



Ανάπτυξη Μεθοδολογίας Σχεδιασμού Βέλτιστων Επεξεργαστών Ειδικού Σκοπού

ΠΕΝΕΔ-2003

Τίτλος:	Καθορισμός του Γενικού Αρχιτεκτονικού Περιγράμματος για Επεξεργαστές Ειδικού Σκοπού Συνόλου Εντολών
Συγγραφείς:	Νικόλαος Καββαδίας (Ερευνητής Α.Π.Θ.)
Κωδικός:	Π 3.1
Έκδοση:	1.0
Τύπος:	Παραδοτέο
Εμπιστευτικότητα:	Δημόσια
Ημερομηνία:	Μάιος 16, 2007
Πρόγραμμα:	Πρόγραμμα Ενίσχυσης του Ερευνητικού Δυναμικού (ΠΕΝΕΔ 2003)
Λέξεις-Κλειδιά:	αρχιτεκτονικό περίγραμμα, επεξεργαστές ειδικού σκοπού, αρχιτεκτονική διαύλου, λειτουργικές μονάδες, μηδενική επιβάρυνση βρόχων, εφαρμογές δοκιμής, εκτίμηση επιδόσεων, πρότυπα εργαλεία ανάπτυξης
Πρόλογος:	Αντικείμενο του παραδοτέου αυτού είναι ο καθορισμός του γενικού αρχιτεκτονικού περιγράμματος των στοχευόμενων επεξεργαστών ειδικού σκοπού (ΕΕΣ) συνόλου εντολών. Για το σκοπό αυτό εξετάζονται, όπως προκύπτουν μέσα από συστηματική διερεύνηση των αντίστοιχων πεδίων λύσεων, οι κατάλληλες επιλογές για την οργάνωση τοπικής μνήμης και καταχωρητών, την αρχιτεκτονική διαύλου, την οργάνωση, χαρακτηριστικά και επικοινωνία/διασύνδεση των λειτουργικών μονάδων των επεξεργαστών, και το σχεδιασμό της μονάδας ελέγχου. Στη διαδικασία αυτή αξιοποιείται η ροή ανάλυσης εφαρμογών της Φάσεως 2, για το χαρακτηρισμό των εφαρμογών δοκιμής για τους ΕΕΣ. Το γενικό αρχιτεκτονικό περίγραμμα των ΕΕΣ θα συμπληρωθεί από τις ειδικές εντολές και λειτουργικές μονάδες που εξυπηρετούν τις εφαρμογές δοκιμής. Οι ειδικές εντολές θα παραχθούν με αυτοποιημένη διαδικασία που περιγράφεται στο παραδοτέο της Φάσης 4.

**Ιστορικό**

Ημερομηνία	Έκδοση	Σχόλια
Μάιος 06, 2007	0.1	Πρώτη πρόχειρη έκδοση
Μάιος 16, 2007	1.0	Τελική έκδοση

Πίνακας περιεχομένων

Πίνακας περιεχομένων	3
1. ΕΙΣΑΓΩΓΗ	4
1.1. Αντικείμενο του παραδοτέου	4
2. ΣΤΟΙΧΕΙΑ ΤΟΥ ΓΕΝΙΚΟΥ ΑΡΧΙΤΕΚΤΟΝΙΚΟΥ ΠΕΡΙΓΡΑΜΜΑΤΟΣ	7
2.1. Σύνολο προκαθορισμένων παραμέτρων του γενικού αρχιτεκτονικού περιγράμματος	7
2.2. Καθορισμός των υποστηριζόμενων RTOS/μικροπυρήνων και επιλογή της υποδομής εξομοίωσης	8
2.3. Διερεύνηση της αρχιτεκτονικής διαύλου	10
2.3.1. Δημοφιλείς αρχιτεκτονικές διαύλου	11
2.4. Διερεύνηση της οργάνωσης της αρχιτεκτονικής μνήμης και καταχωρητών	15
2.5. Καθορισμός των χαρακτηριστικών και της διασυνδεσιμότητας των λειτουργικών μονάδων των ΕΕΣ	20
2.5.1. Καθορισμός των χαρακτηριστικών ιδιοτήτων για την εσωτερική οργάνωση των ειδικών λειτουργικών μονάδων	20
2.5.2. Ένα πλήρες παράδειγμα: Ανάλυση του αλγορίθμου συμπίεσης μορφής κατά MPEG-4 και σχεδιασμός της μονάδας για την λειτουργία «ανεύρεσης διανυσμάτων κίνησης»	21
2.5.3. Καθορισμός της διασυνδεσιμότητας των ειδικών λειτουργικών μονάδων στους ΕΕΣ	26
3. Η ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΕΛΕΓΧΟΥ ΒΡΟΧΩΝ ZOLC	30
3.1. Αναπαράσταση για εφαρμογές με εντατική επεξεργασία βρόχων	31
3.2. Περιπτώσεις χρήσης μιας μηχανής ZOLC	34
3.2.1. Περίπτωση μελέτης: Πολλαπλασιασμός πινάκων κατά μπλοκ σε ένα DSP με λειτουργίες ZOLC	34
3.2.2. Υποστήριξη βρόχων πολλαπλών εισόδων για λειτουργία ZOLC	37
3.2.3. Υποστήριξη για εναλλακτικά μοντέλα λειτουργιών βρόχων στο υλικό (hardware looping)..	39
3.3. Ενσωμάτωση της μονάδας ZOLC σε προγραμματιζόμενους επεξεργαστές	40
3.3.1. Αποψη περιγράμματος ενός επεξεργαστή ενισχυμένου από λειτουργίες ZOLC	40
3.3.2. Αρχιτεκτονικές λεπτομέρειες για τα στοιχεία του ZOLC	41
3.3.3. Προδιαγραφές σε επίπεδο RTL για τους μηχανισμούς ZOLC	43
3.4. Διεξοδική μελέτη των απαιτήσεων για την υποστήριξη του ZOLC στο υλικό και στο λογισμικό	46
3.4.1. Αυτοματοποιώντας τις βελτιστοποιήσεις ZOLC εντός πλαισίου εργασίας μεταγλωττιστή-βελτιστοποιητή επιπέδου συμβολομεταφραστή	46
3.4.2. Αναλύοντας διεξοδικά τις επιπλοκές του ZOLC στο υλικό και στο λογισμικό των ενσωματωμένων επεξεργαστών	49
3.5. Εκτίμηση επιδόσεων για RISC μικροεπεξεργαστές που διαθέτουν μηχανισμούς ZOLC	51
3.5.1. Απαιτήσεις σε υλικό για την υλοποίηση μηχανισμών ZOLC σε ενσωματωμένους επεξεργαστές	51
3.5.2. Επιδόσεις σε κύκλους μηχανής για έναν MIPS-I επεξεργαστή με αρχιτεκτονικές τροποποιήσεις κατά ZOLC	52
3.5.3. Πειράματα για τον hDSP ASIP	55
4. ΤΟ ΓΕΝΙΚΟ ΑΡΧΙΤΕΚΤΟΝΙΚΟ ΠΕΡΙΓΡΑΜΜΑ ΤΩΝ ΕΕΣ	57
5. ΓΛΩΣΣΑΡΙ ΟΡΩΝ	59
6. ΠΑΡΑΡΤΗΜΑ	62
7. ΑΝΑΦΟΡΕΣ	63

1. ΕΙΣΑΓΩΓΗ

Στο πλαίσιο της Φάσης 3, ο ερευνητής Α.Π.Θ. (Νικόλαος Καββαδίας) καθόρισε το γενικό αρχιτεκτονικό περίγραμμα, το οποίο προσδιορίζει τα κατάλληλα χαρακτηριστικά για το ρεπερτόριο εντολών και τη μικροαρχιτεκτονική των επεξεργαστών ειδικού σκοπού (ΕΕΣ) συνόλου εντολών, τους οποίους στοχεύει η μεθοδολογία. Το αρχιτεκτονικό περίγραμμα καθορίζεται από ένα σύνολο παραμέτρων για τα επιμέρους συστατικά στοιχεία που απαρτίζουν έναν επεξεργαστή ΕΕΣ. Το αρχιτεκτονικό περίγραμμα θεωρούμε ότι μπορεί να περιγραφεί από τις εξής συνιστώσες:

1. τον διάδρομο δεδομένων (datapath)
2. τον διάδρομο ελέγχου (control path)
3. την αρχιτεκτονική διαύλου (bus architecture)
4. την οργάνωση μνήμης και καταχωρητών (register and memory organization)

Κάθε μια από τις παραπάνω συνιστώσες μπορεί να διαχωριστεί σε επιμέρους παραμέτρους. Για παράδειγμα, στα πλαίσια της διερεύνησης του πεδίου λύσεων για το ποια είναι η κατάλληλη δομή και ιδιότητες του διαδρόμου δεδομένων, εξετάστηκε η οργάνωση, οι επιμέρους ιδιότητες και ο τρόπος επικοινωνίας/διασύνδεσης των λειτουργικών μονάδων που τον απαρτίζουν και χρησιμοποιούνται για την εκτέλεση των εντολών. Αντίστοιχα, για τη διερεύνηση της κατάλληλης αρχιτεκτονικής διαύλου εξετάστηκαν τα επικρατώντα πρότυπα όσον αφορά χαρακτηριστικές τιμές για τις παραμέτρους τους όπως: οι υποστηριζόμενες τοπολογίες (π.χ. crossbar switch, point-to-point, κ.λ.π.), ο χρονισμός εναλλαγής αφέντη (master), το πρωτόκολλο διαιτησίας (arbitration protocol) καθώς και άλλες.

1.1. Αντικείμενο του παραδοτέου

Στην Φάση 3 διερευνάται το πεδίο λύσεων κάτω από εξωτερικούς περιορισμούς για την εύρεση του γενικού αρχιτεκτονικού περιγράμματος για τους υποστηριζόμενους επεξεργαστές ΕΕΣ. Οι θεωρούμενοι επεξεργαστές ΕΕΣ στο παραδοτέο αυτό (Π3.1) είναι ASIP (Application-Specific Instruction-set Processor) – καθιερωμένος όρος της διεθνούς βιβλιογραφίας – οι οποίοι σχεδιάζονται για τις βέλτιστες επιδόσεις μιας εφαρμογής ή ενός στενού συνόλου εφαρμογών με κοινά χαρακτηριστικά. Η δυνατότητα «αναβάθμισης» στην προοπτική τροποποιήσεων των εφαρμογών στόχων (π.χ. όπως επιβάλλεται από την ανάπτυξη νέων προτύπων, π.χ. όπως συμβαίνει με τη μετάβαση από την απωλεστική συμπίεση στατικής εικόνας κατά JPEG στο πρότυπο JPEG2000) υποστηρίζεται από τη δυνατότητα προγραμματισμού τους. Δηλαδή ο προγραμματιστής-χρήστης ενός ASIP δύναται να προσθέσει νέες λειτουργικότητες σε μορφή ενσωματωμένου λογισμικού (embedded software ως firmware) δεσμευόμενος όμως στη χρήση του ρεπερτορίου εντολών που ήδη διαθέτει ο ASIP. Σε σχέση με τους RISP (Reconfigurable Instruction-set Processor), οι ASIP δεν διαθέτουν τη δυνατότητα επαναπροσδιορισμού (reconfiguration) με την οποία επιτυγχάνεται η τροποποίηση του συνόλου εντολών (άρα και των αντίστοιχων μικροαρχιτεκτονικών χαρακτηριστικών όπως οι λειτουργίες για τις οποίες προσδιορίζονται οι λειτουργικές μονάδες του RISP). Η διευρυνόμενη ευελιξία των RISP έρχεται με κάποιο κόστος, όπως η (σημαντικά) χαμηλότερη μέγιστη συχνότητα ρολογιού, η αυξημένη κατανάλωση ισχύος/ενέργειας και η απαίτηση για μεγαλύτερη επιφάνεια ολοκληρωμένου στις διαφορετικές τεχνολογίες διεργασιών (FPGA, standard-cell ASIC κ.λ.π.).

Στην πράξη, θα πρέπει να σημειωθεί ότι σημαντικό ρόλο στην ώθηση τόσο των ASIP όσο και των RISP διαδραματίζει ο αναμενόμενος χρόνος ζωής των προϊόντων που περιλαμβάνουν

έναν ή παραπάνω ΕΕΣ. Σήμερα, οι αντίστοιχοι χρόνοι είναι ιδιαίτερα σύντομοι (12-18 μήνες για προϊόντα όπως κινητά τηλέφωνα και φορητά τερματικά με δυνατότητες πολυμέσων/gaming) με αποτέλεσμα πολλές φορές το ζητούμενο να είναι η εισαγωγή ενός νέου, βελτιωμένου και προσαρμοσμένου ASIP στην αγορά, το ταχύτερο δυνατό και με ελεγχόμενο κόστος ανάπτυξης/παραγωγής. Για την επίτευξη αυτού του στόχου, απαιτείται το κατάλληλο πλαίσιο εργασίας, το οποίο να μπορεί να προσφέρει:

- ταχεία ανάλυση εφαρμογών, γέννηση/επιλογή ειδικών εντολών, σύνθεση των συναρτησιακών μονάδων και ενσωμάτωση τους στον ASIP
- ταχεία επαναστόχευση (ή γέννηση) των απαιτούμενων εργαλείων ανάπτυξης εφαρμογών (μεταγλωττιστής, συμβολομεταφραστής, συνδέτης, προσομοιωτές σε διαφορετικά επίπεδα ανάλυσης/ακρίβειας, εκσφαλματωτής)

Η ερευνητική προσπάθεια η οποία καταγράφεται στα παραδοτέα των Φάσεων λαμβάνει υπόψη και τις προαναφερθείσες «ρεαλιστικές» απαιτήσεις για την ανάπτυξη επεξεργαστών ειδικού σκοπού συνόλου εντολών.

Στο πλαίσιο της έρευνας κατά τη Φάση 3 του προγράμματος, η διερεύνηση του πεδίου λύσεων για τον καθορισμό του γενικού αρχιτεκτονικού περιγράμματος των ΕΕΣ περιλαμβάνει τα εξής:

Την εξέταση της οργάνωσης της αρχιτεκτονικής μνήμης προγράμματος και δεδομένων και της αρχιτεκτονικής καταχωρητών. Στο μέρος αυτό θα εξεταστεί η κατάλληλη αρχιτεκτονική κρυφής μνήμης (τοπολογία και παράμετροι όπως το μέγεθος, εύρος σειράς, αλγόριθμος ανανέωσης καταχωρήσεων κ.λ.π.) αλλά και η αναγκαιότητα για εξειδικευμένες ιεραρχίες μνήμης. Για την αποδοτική μεταφορά δεδομένων από και προς τα στοιχεία μνήμης, θα διερευνηθεί η χρησιμότητα τεχνικών άμεσης προσπέλασης (DMA). Προκειμένου την εξέταση της οργάνωσης αρχιτεκτονικής καταχωρητών είναι αναγκαίο να εκτιμηθούν παράμετροι που προκύπτουν από τη δυναμική ανάλυση των εφαρμογών δοκιμής (χρόνος ζωής καταχωρητών, κορεσμός καταχωρητών, αριθμός θυρών ανάγνωσης/εγγραφής).

Διερεύνηση των αρχιτεκτονικών διαύλου που: καθορίζονται από πρότυπα (λεπτομερής περιγραφή και τεκμηρίωση) ή/και υφίστανται υλοποιήσεις αναφοράς τους σε γλώσσα σχεδιασμού υλικού (VHDL), συνθέσιμες σε οποιαδήποτε τεχνολογία (FPGA ή standard cell ASIC), κατά προτίμηση δε και σε κατάλληλη μορφή για τη προσομοίωσή τους με ακρίβεια κύκλου (SystemC) και που να είναι διαθέσιμες κάτω από ελεύθερες άδειες. Σκοπός της διερεύνησης αυτής είναι η σύγκριση των αρχιτεκτονικών διαύλου που θα μπορούσαν, χωρίς επιπρόσθετο κόστος, να χρησιμοποιηθούν για την διασύνδεση των ΕΕΣ της μεθοδολογίας σε συστήματα-σε-ολοκληρωμένο (SoC). Ανάμεσα στις αρχιτεκτονικές διαύλου που θα εξεταστούν (ορισμένες έχουν αποτελέσει αντικείμενο αντίστοιχων πρότερων μελετών [Pel03],[Uss01]) ΕΕΣ) είναι οι εξής: AMBA 2.0 (AHB) [AMBA],[Fly97], Coreconnect PLB/OPB [Coreconnect], και Wishbone [Wishbone].

Διερεύνηση της εσωτερικής οργάνωσης (διασύνδεση/επικοινωνία) και του είδους (εσωτερική δομή και εξυπηρετούμενες λειτουργίες) των λειτουργικών μονάδων των ΕΕΣ. Πραγματοποιείται με αποτίμηση των χαρακτηριστικών ποσοτήτων που προκύπτουν από τη στατική και δυναμική ανάλυση των στοχευόμενων εφαρμογών, όπως οι χρησιμοποιούμενοι τύποι δεδομένων, η συχνότητα εμφάνισης εντολών/λειτουργιών (για παράδειγμα αν μια εντολή δεν εμφανίζεται σε καμία από τις εφαρμογές που στοχεύει ο ΕΕΣ, τότε επιτρέπεται η αφαίρεσή της από το υποστηριζόμενο ρεπερτόριο εντολών), κ.α. Η λήψη των χαρακτηριστικών τιμών γίνεται με χρήση (μεταξύ άλλων) των περασμάτων μεταγλωττιστή που περιγράφονται στον Πίνακα 4.2 του παραδοτέου Π2.1. Για παράδειγμα, λαμβάνοντας

υπόψη τα αποτελέσματα για τον μέγιστο και μέσο βαθμό εγγενούς παραλληλίας της εφαρμογής (ILP) θα διερευνηθεί η δυνατότητα για έκδοση πολλαπλών λειτουργιών, με δέσμευση των αντίστοιχων θυρίδων εκτέλεσης στον ίδιο βήμα ελέγχου (control step). Ακόμη, η μελέτη των χρησιμοποιούμενων τύπων δεδομένων μπορεί να αξιοποιηθεί στο σχεδιασμό εντολών τύπου SIMD και των λειτουργικών μονάδων που τις εξυπηρετούν.

Περιπτώσεις συστηματικών παρεμβάσεων στο γενικό αρχιτεκτονικό περίγραμμα. Οι παρεμβάσεις αυτές αντικατοπτρίζουν το ποια ιδιαίτερα χαρακτηριστικά θα πρέπει να υποστηρίζονται σε γενικότερο πλαίσιο για το διάδρομο ελέγχου και το διάδρομο δεδομένων των ΕΕΣ. Τα χαρακτηριστικά αυτά προκύπτουν έπειτα από συνδυασμό της μελέτης των αποτελεσμάτων αλλά και εφόσον υποστηρίζονται από δοκιμές (π.χ. προσομοιώσεις ακρίβειας κύκλου) εμπειρικών εκτιμήσεων του ερευνητή Α.Π.Θ.. Ιδιαίτερα αναφέρουμε την περίπτωση της αρχιτεκτονικής μονάδας ελέγχου βρόχων ZOLC (Zero-Overhead Loop Controller) που έχει αναπτυχθεί από τον ερευνητή Α.Π.Θ. [Καν05a],[Καν06]. Η αρχιτεκτονική ZOLC επιτυγχάνει την εξάλειψη της επιβάρυνσης (σε κύκλους εκτέλεσης) που οφείλεται στην πραγματοποίηση των λειτουργιών βρόχων (όπως αύξηση του δείκτη βρόχου, συνθήκη εξόδου από το βρόχο, και διακλάδωση στην αντίστοιχη διεύθυνση του απαριθμητή προγράμματος), καθώς αυτές εξυπηρετούνται από τον ελεγκτή ZOLC χωρίς την απαίτηση σε κύκλους μηχανής. Μορφές της ZOLC αρχιτεκτονικής ελέγχου δύνανται να χρησιμοποιηθούν τόσο σε μη-προγραμματιζόμενους [Καν05a] όσο και σε προγραμματιζόμενους ΕΕΣ (δηλ. συνόλου εντολών) [Καν06].

Εξέταση του ενδεχόμενου υποστήριξης λειτουργιών διαχείρισης για την επικοινωνία χρήστη-συστήματος. Σημειώνεται ότι γενικά, η ροή σχεδιασμού ΕΕΣ στοχεύει στην ανάπτυξη επεξεργαστών ειδικού σκοπού για την υποστήριξη ενσωματωμένων εφαρμογών, οι οποίες σε πολλές περιπτώσεις δεν απαιτούν τη χρήση λειτουργικού συστήματος. Όταν υπάρχει απαίτηση για χρήση RTOS θα εξετάζεται ο τρόπος εξυπηρέτησης των λειτουργιών διαχείρισης στο SoC σύστημα. Πρέπει να σημειωθεί ότι σημαντικό ρόλο στην ανάπτυξη (σχεδιασμός, εκσφαλμάτωση και αποτίμηση επιδόσεων) λειτουργικών συστημάτων κατέχει η χρήση εξομοιωτή (emulator). Για το λόγο αυτό θα πρέπει να εξεταστούν οι υπάρχουσες υποδομές εξομοίωσης για τις οποίες ο πηγαίος κώδικας είναι ελεύθερα διαθέσιμος υπό μηδενικό κόστος.

Στο τελευταίο στάδιο της Φάσης 3, ο ερευνητής Α.Π.Θ. προβαίνει στην καταγραφή του γενικού αρχιτεκτονικού περιγράμματος. Επίσης, σκιαγραφείται η δομή που προβλέπεται να έχει ένα SoC το οποίο περιλαμβάνει ΕΕΣ της μεθοδολογίας.

Συνοψίζοντας, στα πλαίσια της Φάσης 3 απαιτείται η ικανοποίηση των παρακάτω:

- Καθορισμός της οργάνωσης μνήμης.
- Καθορισμός των προβλεπόμενων αρχιτεκτονικών καταχωρητών.
- Καθορισμός της αρχιτεκτονικής διαύλου για ενσωμάτωση των ΕΕΣ επεξεργασιών σε πρότυπα/υποθετικά SoC.
- Εξαγωγή των χαρακτηριστικών και της διασυνδεσιμότητας των λειτουργικών μονάδων του ΕΕΣ.
- Προσδιορίζονται οι αρχιτεκτονικές ιδιαιτερότητες που επιτρέπεται να περιλαμβάνει ένας ΕΕΣ της μεθοδολογίας.
- Πραγματοποιείται πρώτη εκτίμηση επιδόσεων.
- Καθορισμός των υποστηριζόμενων RTOS/μικροπυρήνων και της υποδομής εξομοιωτή που θα χρησιμοποιηθεί για τη δοκιμή τους.
- Καθορισμός του γενικού αρχιτεκτονικού περιγράμματος για ΕΕΣ.

2. ΣΤΟΙΧΕΙΑ ΤΟΥ ΓΕΝΙΚΟΥ ΑΡΧΙΤΕΚΤΟΝΙΚΟΥ ΠΕΡΙΓΡΑΜΜΑΤΟΣ

2.1. Σύνολο προκαθορισμένων παραμέτρων του γενικού αρχιτεκτονικού περιγράμματος

Στην πράξη, υφίσταται ένα ελάχιστο υποσύνολο χαρακτηριστικών του αρχιτεκτονικού περιγράμματος το οποίο είναι αναγκαίο να προκαθοριστεί ώστε να είναι εφικτή η διαδικασία διερεύνησης των αντίστοιχων πεδίων λύσεων για τις παραμέτρους του. Για το σχεδιασμό του γενικού αρχιτεκτονικού περιγράμματος και κατά επέκταση των στοχευόμενων ΕΕΣ συνόλου εντολών (αντικείμενο της Φάσης 5) δεχόμαστε ότι:

- **το ρεπερτόριο εντολών των ΕΕΣ είτε είναι επεκτάσιμο** (δηλαδή υφίσταται ένα βασικό σύνολο εντολών και οι ειδικές εντολές επιπροστίθενται ως εντολές επέκτασης) **είτε σχεδιάζεται εκ του μηδενός** (designed from scratch)
- **οι ΕΕΣ είναι τύπου RISC**. Η ιδιότητα αυτή δεν απαγορεύει την συνύπαρξη κάποιων «παραδοσιακών» στοιχείων των CISC επεξεργαστών, όπως είναι οι εντολές με εκτέλεση πολλαπλών κύκλων. Τα RISC χαρακτηριστικά τα όποια θεωρούμε από το σημείο αυτό και έπειτα ότι είτε είναι σταθερά είτε καθορίζουν τα όρια των πεδίων λύσεων για όλους τους ΕΕΣ είναι τα εξής:
 - οι εντολές διαθέτουν κωδικοποίηση σταθερού εύρους bit ή εύρος το οποίο είναι ακέραιο πολλαπλάσιο ενός στοιχειώδους εύρους bit
 - οι ΕΕΣ ακολουθούν την αρχιτεκτονική μνήμης Harvard, δηλαδή η μνήμη εντολών είναι διαχωρισμένη από τη μνήμη δεδομένων και δεν χρησιμοποιούν τον ίδιο δίαυλο για την επικοινωνία λέξεων διεύθυνσης και δεδομένων. Οι μνήμες μπορεί να είναι είτε byte-addressable είτε word-addressable ή ακόμη και να έχουν και τις δύο ιδιότητες
 - η αρχιτεκτονική καταχωρητών δεν περιορίζεται στην υλοποίηση ως ομογενές αρχείο καταχωρητών. Οι καταχωρητές που είναι προσβάσιμοι από τον προγραμματιστή (programmer's view) μπορεί να έχουν διαφορετικό εύρος bit
 - επιτρέπονται οι εντολές πολλαπλών κύκλων
 - επιτρέπεται η δρομολόγηση εντολών σε περισσότερες της μιας χρονοθυρίδες (time-slots) και αντίστοιχες λειτουργικές μονάδες
 - επιτρέπεται η ύπαρξη τοπικών διασυνδέσεων για την επικοινωνία μεταξύ των λειτουργικών μονάδων
 - οι ειδικές λειτουργικές μονάδες μπορούν να διαθέτουν στοιχεία τοπικής αποθήκευσης
 - η ανάκληση εντολής από τη μνήμη εντολών γίνεται είτε σε ένα κύκλο είτε σε ένα βήμα ελέγχου
- **οι ΕΕΣ διαθέτουν διεπαφή κατάλληλη για διασύνδεση σε επεξεργαστικό σύστημα μέσω αρχιτεκτονικής διαύλου**. Δηλαδή δεν εξετάζονται οι περιπτώσεις «εξωτικών» μεθόδων επικοινωνίας όπως τα δίκτυα-σε-ολοκληρωμένο (network-on-chip – NoC).
- **Τα εργαλεία ανάπτυξης λογισμικού (software development toolchain) που στοχεύουν τους υποστηριζόμενους επεξεργαστές ΕΕΣ να μπορούν να παραχθούν με τη χρήση ελεύθερου λογισμικού/λογισμικού ανοικτού κώδικα**. Ο περιορισμός αυτός τίθεται για δύο πρακτικούς λόγους: το κόστος απόκτησης της απαιτούμενης υποδομής είναι μηδενικό, και η διαθεσιμότητα του πηγαίου κώδικα

είναι βασική προϋπόθεση για τη δυνατότητα τροποποίησης και επέκτασης του λογισμικού υποδομής κατά απαίτηση.

2.2. Καθορισμός των υποστηριζόμενων RTOS/μικροπυρήνων και επιλογή της υποδομής εξομοίωσης

Στην περίπτωση που απαιτείται η χρήση RTOS σε SoC σύστημα που θα περιλαμβάνει ΕΕΣ της μεθοδολογίας, θεωρείται ότι υπάρχουν δύο βασικές δυνατότητες:

- το σύστημα να περιλαμβάνει έναν GPP με RISC αρχιτεκτονική που θα πραγματοποιεί τις λειτουργίες ελέγχου και διαχείρισης. Ο GPP αναλαμβάνει όλη την επικοινωνία και γνωστοποιεί στον επεξεργαστή ειδικού σκοπού (με κάποιου είδους κλήση συνάρτησης ή θεωρώντας τον ως περιφερειακό χαρτογραφημένο στην μνήμη) ότι πρέπει να εκτελέσει κάποια διεργασία που απαιτεί η ενσωματωμένη εφαρμογή.
- Το RTOS εξυπηρετείται από τον ίδιο τον ΕΕΣ. Στην περίπτωση που διαθέτει ικανό βασικό σύνολο εντολών, οι δυνατότητες εστιάζονται στο να υποστηρίζεται εξειδικευμένο λειτουργικό σύστημα από τον ΕΕΣ, το οποίο δεν προϋποθέτει την ύπαρξη μονάδας διαχείρισης μνήμης (MMU). Στην περίπτωση αυτή, δεν χρησιμοποιείται εικονική μνήμη (virtual memory). Θεωρούμε ότι αυτή είναι μια εφικτή λύση, καθώς περιορίζει σημαντικά την πολυπλοκότητα του ζητήματος. Άλλη ενδεχόμενη λύση είναι η στόχευση κάποιου απλού μικροπυρήνα (microkernel) ο οποίος να μην προϋποθέτει ούτε καν την υποστήριξη συστήματος αρχείων (file system).

Η απόφαση για την υποστήριξη ή όχι λειτουργικού συστήματος, δεν επηρεάζει άμεσα το αρχιτεκτονικό περίγραμμα του ίδιου του επεξεργαστή ειδικού σκοπού, αλλά το σύστημα (SoC) στο οποίο χρησιμοποιείται. Για παράδειγμα, όταν υπάρχει απαίτηση υποστήριξης RTOS για επεξεργαστές αρχιτεκτονικής MIPS-I (ή παραπλήσιας) είναι αναγκαία η προσθήκη του συνεπεξεργαστή συστήματος (cop0) για την υποστήριξη διακοπών (interrupts) και εξαιρέσεων λογισμικού (software exceptions).

Τα λειτουργικά συστήματα (RTOS/μικροπυρήνες) προς εξέταση καθορίστηκε να ικανοποιούν κατ' αρχήν τα εξής:

- να έχουν ήδη στοχευτεί σε επεξεργαστές MIPS-I ή τύπου DLX. Θεωρούμε ότι το ρεπερτόριο εντολών των δύο αυτών αρχιτεκτονικών είναι πολύ κοντά στην εικονική μηχανή SUIFvm, που περιγράφει μια γενική και αντιπροσωπευτική RISC αρχιτεκτονική, αρκετά απλή ώστε να θεωρηθεί ως το βασικό σύνολο εντολών ενός ΕΕΣ. Αυτή η συνθήκη εξασφαλίζει ότι ο κόπος για την επαναστόχευσή τους (porting/retargeting) για τον ΕΕΣ θα είναι ελεγχόμενος καθώς οι δύο προαναφερθείσες αρχιτεκτονικές είναι απλούστερα οργανωμένες και διαθέτουν ικανοποιητική τεκμηρίωση.
- αν δεν συμβαίνει το παραπάνω, τότε θα πρέπει τα RTOS να είναι άμεσα επαναστοχεύσιμα σε οποιανδήποτε αρχιτεκτονική (με ενδεχόμενες πολύ περιορισμένες ρυθμίσεις, όπως για συγκεκριμένες διευθύνσεις στο χάρτη μνήμης)
- να έχει δοκιμαστεί ή να υπάρχει ικανή τεκμηρίωση για αυτό σε συνδυασμό με υπάρχοντα εργαλεία προσομοίωσης (σε SystemC) ή/και εξομοίωσης

Έτσι, τα λειτουργικά συστήματα τέθηκαν υπ' όψη είναι τα εξής:

- uClinux χωρίς ή με υποστήριξη MMU (για paging και multi-tasking). Αρκετά δημοφιλής είναι η βασική διανομή του uClinux [uClinux].
- Topsy, ένα εκπαιδευτικό OS για τον MIPS.

- DLXOS, εκπαιδευτικό OS για τον DLX.
- Το λειτουργικό σύστημα μηχανής πεπερασμένων καταστάσεων (FSMOS) NESOS που είναι πλήρως υλοποιημένο σε ANSI C. Αυτή η επιλογή είναι σχεδόν επιτακτική στην περίπτωση απουσίας GPP στο SoC, καθώς επιτρέπει την άμεση προτυποποίηση του OS, ανεξάρτητα του ακριβούς συνόλου εντολών και χαρακτηριστικών του ΕΕΣ.
- Plasma RTOS [Mlite], το δοκιμασμένο στην πράξη RTOS που διατίθεται μαζί με τη διανομή του Mlite soft-core επεξεργαστή. Το RTOS αυτό έχει σχεδιαστεί για χρήση σε εφαρμογές δικτύου καθώς διαθέτει υλοποίηση της TCP/IP στοίβας και HTTP server. Προσόν του είναι επίσης οι υλοποιήσεις μέρους των βιβλιοθηκών libc και libm που μπορούν να χρησιμοποιηθούν και σε άλλες χρήσεις του Plasma RTOS.
- Το TinyOS που σχεδιάστηκε κυρίως για χρήση σε συστήματα αισθητήρων.

Γενικά, η κατάλληλη επιλογή για το RTOS καθορίζεται σε συνάρτηση με τις απαιτήσεις των εφαρμογών που πρόκειται να εξυπηρετούνται στο σύστημα που περιλαμβάνει τον ΕΕΣ. Από τα παραπάνω λειτουργικά, το uClinux (χωρίς MMU) είναι μια καλή πρώτη επιλογή καθώς πρόκειται για ένα OS (αν και όχι ακριβώς πραγματικού χρόνου) που έχει σήμερα κάθε πιθανή χρήση (ενώ π.χ. το Plasma RTOS και το TinyOS είναι περισσότερο στοχευμένα) και είναι ένα από τα περισσότερο διαδεδομένα. Κατά δεύτερο λόγο, χρήσιμα είναι και τα NESOS και Plasma RTOS, καθώς προσφέρουν εύκολα επαναχρησιμοποιήσιμο κώδικα για την ανάπτυξη ενός ελαφρώς τροποποιημένου RTOS, πιο κοντά στις εκάστοτε ανάγκες.

Πρέπει να σημειωθεί ότι σημαντικό ρόλο στην ανάπτυξη (σχεδιασμός, εκσφαλμάτωση και αποτίμηση επιδόσεων) RTOS κατέχει η χρήση εξομοιωτή (emulator). Ο εξομοιωτής (όπως νοείται εδώ) παρέχει τα εξής:

- Προσομοιωτή ακρίβειας εντολής για τον στοχευόμενο επεξεργαστή (είτε για τον GPP που οδηγεί τον ΕΕΣ είτε για τον ίδιο τον ΕΕΣ στην περίπτωση που εκτελεί λειτουργίες διαχείρισης συστήματος)
- Υλοποιήσεις συσκευών σε επίπεδο συστήματος όπως τερματικά (terminals), ελεγκτές θυρών (RS232, PCI, USB κ.λ.π.), ελεγκτές συσκευών (δίσκων IDE, μνημών flash, CDROM), δικτύου (Ethernet), DMA, μετρητών (timers) κ.λ.π.
- Υπάρχουν διαθέσιμες (ή μπορούν να συνταχθούν από τον ερευνητή Α.Π.Θ. τροποποιώντας ήδη υπάρχουσες για αρχιτεκτονικές MIPS-I ή DLX) διανομές των λειτουργικών συστημάτων οι οποίες θα πρέπει να μπορούν να προσομοιωθούν με χρήση της υποδομής εξομοίωσης.

Οι υποδομές εξομοίωσης για τις οποίες ο πηγαίος κώδικας είναι ελεύθερα διαθέσιμος υπό μηδενικό κόστος και είναι εντός προδιαγραφών είναι οι: Gxemul [GXemul], qemu [Qemu], και vmips [VMIPS].

GXemul [GXemul]

Ο GXemul είναι ένας αρκετά πλήρης εξομοιωτής επεξεργαστικών συστημάτων με ακρίβεια εντολής. Είναι εξ ολοκλήρου γραμμένος σε C, και διαθέτει ικανοποιητικές υλοποιήσεις των επεξεργαστών MIPS, PowerPC, ARM και SuperH ενώ αναφέρεται ότι ορισμένοι συνδυασμοί (κύριως όσον αφορά τον MIPS) επεξεργαστή-λειτουργικού συστήματος λειτουργούν πλήρως. Για παράδειγμα ο GXemul εξομοιώνει χωρίς προβλήματα το NetBSD πάνω σε MIPS συστήματα. Επίσης, η διανομή του GXemul παρέχει ένα βασικό σύστημα (test-machine) το οποίο μπορεί να χρησιμοποιηθεί ως βάση για την ανάπτυξη συστημάτων για τους ΕΕΣ.

Ο GXemu διαθέτει ολοένα και αυξανόμενη βάση χρηστών, αλλά κάποια επιθυμητά χαρακτηριστικά δεν υφίστανται, όπως είναι η δυνατότητα συνεργιστικής λειτουργίας με SystemC προσομοιωτές (επεξεργαστών ή/και περιφερειακών).

Qemu [Qemu]

Ο Qemu αποτελεί μια διαδεδομένη υποδομή εξομοιωτή και διαθέτει ίσως τη μεγαλύτερη βάση χρηστών. Υποστηρίζει την εκτενέστερη γκάμα επεξεργαστών (IA-32, AMD64, MIPS, SPARC, ARM, SH4, PowerPC) και περιφερειακών (USB ελεγκτές, κάρτες δικτύου, κ.λ.π.). Το κυριότερο πλεονέκτημά του είναι ότι έχει αναπτυχθεί κατάλληλο εργαλείο για την ενσωμάτωση επεξεργαστών και περιφερειακών που έχουν συνταχθεί σε SystemC, αλλά και υλοποίηση της αρχιτεκτονικής διαύλου AMBA (ενότητα 2.3).

VMIPS [VMIPS]

Ο VMIPS αποτελεί έναν εξομοιωτή που έχει αναπτυχθεί για την MIPS R3000 RISC αρχιτεκτονική και υλοποιεί ένα βασικό υπολογιστικό σύστημα. Οι διαθέσιμες συσκευές είναι μάλλον περιορισμένες, καθώς πρόκειται για μια απλή κονσόλα, ελεγκτή πληκτρολογίου (PS/2), και ορισμένες συσκευές του DECstation 5000. Το σημαντικότερο όμως είναι ότι ο συντάκτης του VMIPS δεν παρέχει ολοκληρωμένα παραδείγματα εξομοίωσης κάποιου OS.

Λαμβάνοντας υπ' όψη τα παραπάνω, είναι φανερό ότι η **υποδομή Qemu** με τις προσθήκες συνεργιστικής υποστήριξης αρθρωμάτων (modules) σε SystemC **είναι η πιο πρόσφορη επιλογή**.

2.3. Διερεύνηση της αρχιτεκτονικής διαύλου

Στο πλαίσιο της διερεύνησης για το γενικό αρχιτεκτονικό περίγραμμα των επεξεργαστών ΕΕΣ είναι αναγκαίο να καθοριστεί και η διεπαφή του (ο τρόπος επικοινωνίας/διασύνδεσης) στο πλαίσιο του SoC συστήματος στο οποίο χρησιμοποιείται. Οι δύο επικρατώντες τρόποι διασύνδεσης SoC είναι η αρχιτεκτονική διαύλου (bus architecture) και το δίκτυο-σε-ολοκληρωμένο (NoC). Για τους ΕΕΣ της μεθοδολογίας θεωρούμε ότι πρόκειται κατ' αρχήν να επικοινωνούν μέσω αρχιτεκτονικής διαύλου για τους εξής λόγους:

- Από σχετικές μελέτες (αν και σαφώς υπάρχουν «φανατικοί» που υπερθεματίζουν με αφοσίωση υπέρ του ενός ή του άλλου τρόπου διασύνδεσης) είναι σαφές ότι τα NoC απευθύνονται σε αρκετά πολύπλοκα συστήματα (με αρκετές δεκάδες ή εκατοντάδες στοιχεία επεξεργασίας). Μάλιστα, καθώς η επικοινωνία γίνεται μέσω πακέτων, η επιβάρυνση κύκλων για την διεκπεραίωση των μεταβιβάσεων μηνυμάτων σε ένα NoC ανέρχεται συνήθως σε δεκάδες κύκλων. Τα SoC που θεωρούνται στα πλαίσια του προγράμματος, υποθέτουμε ότι συντίθενται από μικρό αριθμό στοιχείων επεξεργασίας (τυπικά 2-20) με απαίτηση ταχείας μεταβίβασης δεδομένων, διευθύνσεων και σημάτων ελέγχου/χειραψίας.
- Υπάρχουν καλώς τεκμηριωμένα και δοκιμασμένα (δηλ. βρίσκονται σε χρήση για ικανό χρονικό διάστημα) πρότυπα (προδιαγραφές) για αρχιτεκτονικές διαύλου. Στον χώρο των NoC δεν έχουν διαμορφωθεί πρότυπα επικυρωμένα από κάποια ανεξάρτητη αρχή ή κονσόρτιο και τα οποία να βρίσκονται σε εκτενή χρήση.
- Για τα περισσότερα πρότυπα αρχιτεκτονικών διαύλου υφίστανται ελεύθερα διαθέσιμες υλοποιήσεις αναφοράς. Και πάλι αυτό δεν συμβαίνει για τα NoC. Αντίθετα, η ανάπτυξη για NoC στηρίζεται σε υποδομές προσομοίωσης (π.χ. NoCem [NoCem], Noxim [Noxim]), οι περισσότερες από τις οποίες βρίσκονται σε

δοκιμαστικό στάδιο.

Στην υποενότητα αυτή θα εξεταστούν λύσεις αρχιτεκτονικής διαύλου οι οποίες ικανοποιούν κάποιες βασικές προϋποθέσεις:

- Διατίθενται υπό άδεια που εξασφαλίζει τη δωρεάν χρήση τους (εξασφάλιση της μη καταβολής οποιασδήποτε μορφής royalties).
- Υιοθετούν σύγχρονο χρονισμό.
- Είναι υλοποιήσιμα σε οιαδήποτε τεχνολογία ολοκληρωμένων κυκλωμάτων (FPGA, ASIC).
- Συνοδεύονται από ικανή τεκμηρίωση.
- Υφίστανται υλοποιήσεις αναφοράς κατά προτίμηση σε VHDL (για σύνθεση σε υλικό) και σε SystemC, σε επίπεδο RTL ή σε μορφή μοντέλου επιπέδου συναλλαγής (Transaction Level Model - TLM).
- Υπάρχουν διαθέσιμες (ή είναι εύκολο να συνταχθούν) γέφυρες (bridges) για τη διεπαφή με άλλα πρότυπα αρχιτεκτονικών διαύλου. Εφόσον αυτό συμβαίνει, IP πυρήνες (IP cores) που είχαν αρχικά σχεδιαστεί για αυτά τα πρότυπα μπορούν να χρησιμοποιηθούν (με χρήση κάποιας ενδιάμεσης λογικής) και για το επιλεγμένο πρότυπο.

2.3.1. Δημοφιλείς αρχιτεκτονικές διαύλου

Οι αρχιτεκτονικές διαύλου που θα εξεταστούν είναι οι: AMBA 2.0 AHB [Fly97],[AMBA] και η απλοποιημένη μορφή του AHB-Lite [AHB-Lite], Coreconnect PLB/OPB [CoreConnect], Wishbone [Wishbone], SimpCon [SimpCon], Simplebus [Simplebus], και Greenbus [Greenbus].

AMBA [Fly97],[AMBA],[ARM]

Η προδιαγραφή αρχιτεκτονικής διαύλου AMBA (Advanced Microcontroller Bus Architecture) αναπτύχθηκε από την ARM [Fly97],[ARM] και καθορίζει ένα σύνολο προτύπων για το σχεδιασμό ενσωματωμένων επεξεργαστικών συστημάτων υψηλών επιδόσεων. Η προδιαγραφή AMBA βρίσκεται υπό συνεχή εξέλιξη. Οι περισσότεροι δημοφιλείς εκδοχές είναι η AMBA 2.0 και η νεώτερη AMBA AXI.

Το πρότυπο AMBA 2.0 καθορίζει τρεις διαφορετικές αρχιτεκτονικές διαύλων, τις AHB (Advanced High-Performance Bus), APB (Advanced Peripheral Bus) και ASB (Advanced System Bus). Η APB προσφέρει σημαντικά χαμηλότερες επιδόσεις και αρχικά προοριζόταν μόνο για τη διασύνδεση ελεγκτών περιφερειακών συσκευών μέσω γέφυρας (arb2ahb), ενώ το ASB είναι στην ουσία μια παλαιότερη εκδοχή του AHB. Έτσι, μόνο το πρότυπο AHB (το APB ενδέχεται να χρησιμοποιηθεί για ορισμένους ελεγκτές συσκευών) έχει ουσιαστικό ενδιαφέρον.

Το πρότυπο AMBA AXI (v 1.0) προσθέτει επιπλέον δυνατότητες στα AHB και APB, ενώ είναι προς τα πίσω συμβατό (backward compatible) με αυτά. Οι κυριότερες από τις δυνατότητες αυτές είναι: ολοκλήρωση μεταβίβασης εκτός-σειράς (out-of-order), καλύτερη υποστήριξη αφεντών (masters) τύπου DMA, ξεχωριστές φάσεις διευθυνσιοδότησης/ελέγχου και δεδομένων, απλοποιημένες μεταβιβάσεις ριπής (burst transfers) και υποστήριξη για μη ευθυγραμμισμένες μεταβιβάσεις δεδομένων (unaligned transfers).

Η μεγάλη διάδοση του AMBA AHB, καθώς υφίσταται μεγάλος αριθμός από παραδείγματα περιφερειακών [Opencores],[ARM] αλλά και ορισμένα εργαλεία αυτοματοποίησης μέρους του σχεδιασμού της αρχιτεκτονικής διαύλου (όπως το λογισμικό ahb_system_generator [ASG]), αλλά και η ύπαρξη υλοποιήσεων αναφοράς τόσο σε VHDL όσο και σε SystemC, το καθιστούν ικανοποιητική επιλογή.

AHB-Lite [AHB-Lite]

Το πρότυπο AHB-Lite αποτελεί υποσύνολο της πλήρους AMBA 2.0 AHB προδιαγραφής. Το AHB-Lite καθορίζει την αρχιτεκτονική διαύλου για συστήματα μονού αφέντη (single-master). Όσον αφορά AHB συστήματα με ιεραρχία πολλαπλών στρώσεων (multi-layered) προβλέπεται η ύπαρξη ενός master ανά επίπεδο.

Αφέντες και σκλάβοι (slaves) μπορούν να ανταλλαχθούν εύκολα ανάμεσα σε ένα πλήρες AHB και ένα AHB-Lite σύστημα. Οι μόνες περιπτώσεις που απαιτούν λογική περιτυλίγματος (wrapper) είναι η χρήση ενός AHB-Lite master σε AHB σύστημα και ενός AHB slave (ο οποίος δεν διαθέτει τα σήματα διεπαφής Split/Retry) στο AHB-Lite.

Στην ουσία, το AHB-Lite προσφέρει μία απλούστερη επιλογή, απολύτως συμβατή με το AHB, για συστήματα single-master, διατηρώντας τα χαρακτηριστικά των διαύλων AHB.

Coreconnect PLB/OPB [CoreConnect]

Το πρότυπο Coreconnect [CoreConnect] περιλαμβάνει τις προδιαγραφές PLB (Processor Local Bus), OPB (On-chip Peripheral Bus) και DCR bus (Device Control Register bus) και περιγράφει μια ιεραρχικά δομημένη αρχιτεκτονική διαύλου που αναπτύχθηκε από την IBM. Το πρότυπο PLB (αντίστοιχο από πολλές απόψεις του AMBA AHB) υποστηρίζει οργάνωση διαύλου πολλαπλών master – πολλαπλών slave με κεντρική διαίτησία (central arbitration). Ο δίαυλος OPB (κατά το APB) επιτρέπει πολλαπλούς master – πολλαπλούς slave αλλά οι μεταβιβάσεις είναι αποκλειστικά μονού κύκλου (single-cycle) και υλοποιείται μόνο με κατανεμημένη πολύπλεξη.

Σήμερα, φαίνεται ότι από τα δύο κύρια πρότυπα CoreConnect (PLB, OPB) χρησιμοποιείται περισσότερο ο δίαυλος OPB (αν και τεχνικά κατώτερος) καθώς έχει επιλεγεί από την Xilinx [Xilinx] ως το πρότυπο για την διασύνδεση συστημάτων που κάνουν χρήση του ιδιοκτησιακού επεξεργαστή MicroBlaze [MicroBlaze]. Επίσης, ο σχεδιασμός γεφυρών από το OPB σε AHB και το αντίστροφο δεν είναι τριτομμένη διαδικασία. Για ορισμένα Xilinx FPGA (VirtexII-Pro, Virtex-5) που διαθέτουν προκαθορισμένο-στο-υλικό (hard-core) επεξεργαστή PowerPC, η διασύνδεση των τοπικών περιφερειακών του PowerPC γίνεται κατά το πρότυπο PLB. Το πρόβλημα με αρκετά από τα περιφερειακά (ελεγκτές) που διατίθενται από την Xilinx είναι ότι είναι συνθέσιμα αποκλειστικά σε ιδιοκτησιακά FPGA.

Γενικά, τα πρότυπα CoreConnect θεωρούνται αρκετά πλήρη όσον αφορά το κομμάτι των προδιαγραφών και της τεκμηρίωσης. Ένα μειονέκτημα είναι ότι είναι περισσότερο πολύπλοκα από τα αντίστοιχα του AMBA και η υλοποίηση αναφοράς είναι σε Verilog.

Wishbone [Wishbone]

Η αρχιτεκτονική SoC διασύνδεσης Wishbone αποτελεί μια μεταφερτή διεπαφή που μπορεί να χρησιμοποιηθεί για την διασύνδεση IP μονάδων με οργάνωση διαύλου. Οι προδιαγραφές Wishbone προβλέπουν ποικιλία τρόπων διασύνδεσης όπως σημείο-σε-σημείο (point-to-point), ροής δεδομένων (data flow), διαμοιρασμένου διαύλου (shared bus) και crossbar switch. Ιδιαίτερα, φαίνεται πως πρόκειται για το μοναδικό πρότυπο που προσδιορίζει την οργάνωση διαύλου κατά τη ροή δεδομένων (σύνδεση των IP κατά τη ροή προώθησης των δεδομένων όπως θα συνέβαινε σε μια γραμμή παραγωγής), ο οποίος είναι φυσικός τρόπος για τη διασύνδεση απλών μη-προγραμματιζόμενων μονάδων. Εισάγει επίσης την έννοια των πινακίδων (tags) για τη δευτερεύουσα επισήμανση κάποιων σημάτων.

Κάτι όμως που κάνει τις προδιαγραφές Wishbone αρκετά διαφορετικές από τις AMBA και Coreconnect είναι το γεγονός ότι προσφέρουν μεγαλύτερο βαθμό ελευθερίας στο σχεδιασμό διασυνδέσεων μεταξύ αρθρωμάτων υλικού (modules) που σημαίνει ότι λιγότερες παράμετροι είναι οριστικοποιημένες από το πρότυπο. Συνοπτικά, το Wishbone ορίζει μια σειρά από ΚΑΝΟΝΕΣ (RULES), ΣΥΣΤΑΣΕΙΣ (RECOMMENDATIONS), ΕΙΣΗΓΗΣΕΙΣ (SUGGESTIONS), ΑΔΕΙΕΣ (PERMISSIONS) και ΠΑΡΑΤΗΡΗΣΕΙΣ (OBSERVATIONS). Αυτό δημιουργεί το

πρόβλημα ενδεχόμενων ασυμβατοτήτων (incompatibilities) μεταξύ των διαύλων Wishbone και των αντίστοιχων αρθρωμάτων που εξυπηρετούν. Επίσης, αυτός ο παράγοντας αυξάνει τη δυσκολία στην ανάπτυξη γενικής χρήσης γεφυρών και περιτυλιγμάτων για επαναχρησιμοποίηση Wishbone IPs σε άλλους διαύλους και το αντίστροφο.

Το Wishbone βρίσκεται στο δημόσιο πεδίο (Public Domain) κάτι που εξασφαλίζει την απρόσκοπτη και χωρίς καμία υποχρέωση/δέσμευση χρήση του. Επίσης σήμερα υπάρχει πλήθος παραδειγμάτων μονάδων σύνθεσης του διαύλου (κυκλώματα διαιτησίας, αποκωδικοποιητές διευθύνσεων), επεξεργαστών, και περιφερειακών [Wishbone],[Opencores] που ακολουθούν το πρότυπο Wishbone, με βασικότερο παράδειγμα τον ανοικτό επεξεργαστή OpenRISC [OpenRISC] και τα SoC συστήματα που τον χρησιμοποιούν (OR1200 κ.λ.π.).

SimpCon [SimpCon]

Η διασύνδεση SimpCon [SimpCon] αποτελεί μία από τις απλούστερες υπάρχουσες αρχιτεκτονικές διαύλου και ακολουθεί το πρότυπο σημείο-σε-σημείο μεταξύ ενός master και ενός slave. Αναπτύχθηκε από τον Martin Schoeberl στο περιθώριο του επιτυχημένου JOP (Java-Optimized Processor) και του SoC αναφοράς του [JOP] ως μια βελτίωση ενός υποσυνόλου του Wishbone. Σκοπός του SimpCon είναι η αντιμετώπιση δύο βασικών προβλημάτων (ή μάλλον όχι τόσο καλών χαρακτηριστικών) που είναι κοινός τόπος σε όλες τις αρχιτεκτονικές διαύλου που προέρχονται ιστορικά από προδιαγραφές διαύλου σε επίπεδο off-chip (board-level) όπως είναι τα AMBA και Wishbone. Συγκεκριμένα, αυτά είναι ότι οι slaves πρέπει να διατηρούν έγκυρα δεδομένα (αποτελέσματα) μόνο για ένα κύκλο, ενώ οι master πρέπει πάντα να διατηρούν τα δεδομένα έγκυρα από την πλευρά τους σε όλη τη διάρκεια της μεταβίβασης, ακόμη και αν αυτή διαρκεί πάνω από ένα κύκλο και τα δεδομένα αυτά έχουν ήδη διαβαστεί.

Αν και κάποιες παράμετροι (όπως ο χειρισμός των διακοπών) δεν καλύπτονται από τις προδιαγραφές, το SimpCon έχει αρκετά πλεονεκτήματα: η διανομή (ανοικτής άδειας) περιλαμβάνει υλοποιήσεις αναφοράς για αρκετά περιφερειακά, ενώ παρέχονται γέφυρες για τα πρότυπα AHB και κλασσικές υλοποιήσεις του Wishbone.

Simplebus [Simplebus]

Το πρότυπο Simplebus [Simplebus] διαφέρει σημαντικά από τα προηγούμενα. Πρόκειται για προδιαγραφές που υφίστανται σε αρκετά υψηλότερο επίπεδο αφαίρεσης (cycle-accurate TLM) από όλους τους προαναφερθέντες διαύλους, και αναπτύχθηκε ως ένα παράδειγμα υλοποίησης προσομοιωτών συστημάτων σε SystemC [SystemC]. Η απλότητα του Simplebus δεν επιτρέπει την υποστήριξη για μεταβιβάσεις με διοχέτευση, διαχωρίσιμες μεταβιβάσεις, πρωτόκολλο χειραψίας αλλά και άλλων χαρακτηριστικών που είναι κοινά στους περισσότερους από τους προηγούμενους διαύλους. Αντίθετα, προσφέρεται ως εργαλείο για την άμεση και γρήγορη προτυποποίηση (rapid prototyping) συστημάτων στη γλώσσα SystemC, κάτι που συνήθως είναι επιθυμητό στα πρώτα στάδια της ροής σχεδιασμού ΕΕΣ (αλλά και μπορεί να χρησιμοποιείται ως η υλοποίηση αναφοράς και να συγκρίνεται με τη συμπεριφορά της VHDL του συστήματος του ΕΕΣ).

GreenBus [GreenBus]

Η προδιαγραφή GreenBus [GreenBus] έχει σημαντικές ομοιότητες με την Simplebus όμως προσφέρει περισσότερες δυνατότητες για τη μοντελοποίηση διαύλου/διαύλων σε SoC συστήματα στη γλώσσα SystemC. Στην ουσία καθορίζει ένα πλαίσιο εργασίας για την γρήγορη προτυποποίηση αρχιτεκτονικών διαύλου σε SystemC και δεν πρόκειται για αυστηρή προδιαγραφή κάποιας συγκεκριμένης αρχιτεκτονικής διαύλου. Είναι σημαντικό ότι υλοποιήσεις του GreenBus μπορεί να υφίστανται τόσο σε επίπεδο TLM (μια τέτοια περιγραφή δεν είναι συνθέσιμη σε υλικό) όσο και σε RTL (που θεωρητικά είναι μετατρέψιμη από SystemC σε VHDL).

Πίνακας 2.1: Περίληψη των αρχιτεκτονικών διαύλου που είναι δωρεάν διαθέσιμοι

Αρχιτεκτονική διαύλου	Τοπολογία					Διασύνδεση		Πρωτόκολλο διαίτησης	Εύρος bit		Πρωτόκολλο μεταβίβασης				Άδεια χρήσης	Τεκμηρίωση (1-5)	Υλοποιήσεις αναφοράς	Δυσκολία χρήσης/κατανόησης (1-5)
	Σημείο-σε-σημείο	Δακτύλιος	Διαμοιρασμένη ενός επιπέδου	Ιεραρχική	Δίκτυο διασύνδεσης crossbar	Κατευθυντικότητα	Επικοινωνία		Λέξη δεδομένων	Λέξη διεύθυνσης	Χειραψία	Διαχωρισμένη	Διοχετευμένη	Τρόποι μεταβίβασης				
AMBA AHB 2.0	—	—	—	■	—	M	Π	Κατά περίπτωση	32,64,128,256	32	■	■	■	Single R/W Burst (4,8,16) B,H,W transfer	Δεσμευτική άδεια	5	VHDL SystemC (QEMU,RTL)	3
AMBA AHB-Lite	—	—	—	■	—	M	—	—	32,64,128,256	32	■	■	■	όπως παραπάνω	Δεσμευτική άδεια			
CoreConnect PLB/OPB	—	■	—	■	—	M	Π	Κατά περίπτωση	32,64,128,256	32	■	■	■	Single R/W Overlapped R/W Burst (16-64)	Δεσμευτική άδεια	5	Verilog	4
Wishbone	■	■	■	—	■	MΔ	ΠΤ	Κατά περίπτωση	8-64	8-64	■	—	—	Single R/W Block transfer RMW Event	Public domain	5	VHDL	2
SimpCon	■	—	—	—	—	M	Π	—	32	32	—	—	■	Single R/W	Public domain	3	VHDL	1
Simplebus	■	—	■	■	—	MΔ	—	Κατά περίπτωση	N/A	N/A	—	—	—	Single R/W Burst R/W	SystemC OSL	3	SystemC (TLM)	2
GreenBus	■	—	■	■	—	MΔ	N/A	Κατά περίπτωση	N/A	N/A	■	N/A	N/A	Single R/W Burst R/W	N/A	4	SystemC (TLM,RTL)	2

Υπόμνημα:

Συντόμευση	Σημασία
Δ	Διπλός/Διπλή
M	Μονός/Μονή
Π	Πολυπλέκτες
Ta	Τρισταθείς πύλες
R/W	Ανάγνωση/Εγγραφή
R/M/W	Ανάγνωση/Τροποποίηση/Εγγραφή
N/A	Μη εφαρμόσιμος όρος/Άγνωστο

Η απόφαση για την κατάλληλη αρχιτεκτονική διαύλου στην πραγματικότητα δεν είναι μονοσήμαντη. Δεν είναι άλλωστε καθόλου σπάνιο να συντίθενται συστήματα για κάποιο μεν πρότυπο διαύλου αλλά να ενσωματώνουν δε στοιχεία με διεπαφές για άλλους διαύλους διαμέσου κατάλληλων γεφυρών.

Οι παράγοντες που σταθμίζονται στην τελική επιλογή αρχιτεκτονικής διαύλου είναι:

- Τα *τεχνικά χαρακτηριστικά* (τοπολογία, πρωτόκολλο διαιτησίας, πρωτόκολλο μεταβίβασης)
- Το *πλαίσιο χρήσης* (διαθέσιμη τεκμηρίωση, υλοποιήσεις αναφοράς, ενσωμάτωση σε υφιστάμενες υποδομές προσομοίωσης/εξομοίωσης)
- *Διαθεσιμότητα υπαρχόντων IPs* (προκειμένου την επαναχρησιμοποίησή τους) όπως γέφυρες διαύλων, λογική περιτυλίγματος, ελεγκτές περιφερειακών συσκευών κ.α.

Έπειτα από διεξοδική εξέταση των παραπάνω δεδομένων **αποφασίστηκε η επιλογή της αρχιτεκτονικής διαύλου AMBA 2.0 AHB** ως ο βασικός τρόπος οργάνωσης της διασύνδεσης των επιμέρους στοιχείων σε επίπεδο SoC. Σημειώνονται επίσης τα εξής:

- Για SoC με ένα master, επιτρέπεται η χρήση της απλοποιημένης μορφής AHB-Lite. Αυτό προϋποθέτει την μετατροπή της υλοποίησης αναφοράς του AHB-Lite από Verilog σε VHDL (Φάση 5).
- Επιτρέπεται η χρήση IPs συμβατών με άλλους διαύλους (όπως με Wishbone και SimpCon ή ακόμη και Avalon [Altera] εφόσον το αντίστοιχο IP δεν καλύπτεται από royalties), καθώς υφίστανται διαθέσιμα bridges και wrappers για την ενσωμάτωσή τους σε συστήματα AMBA.

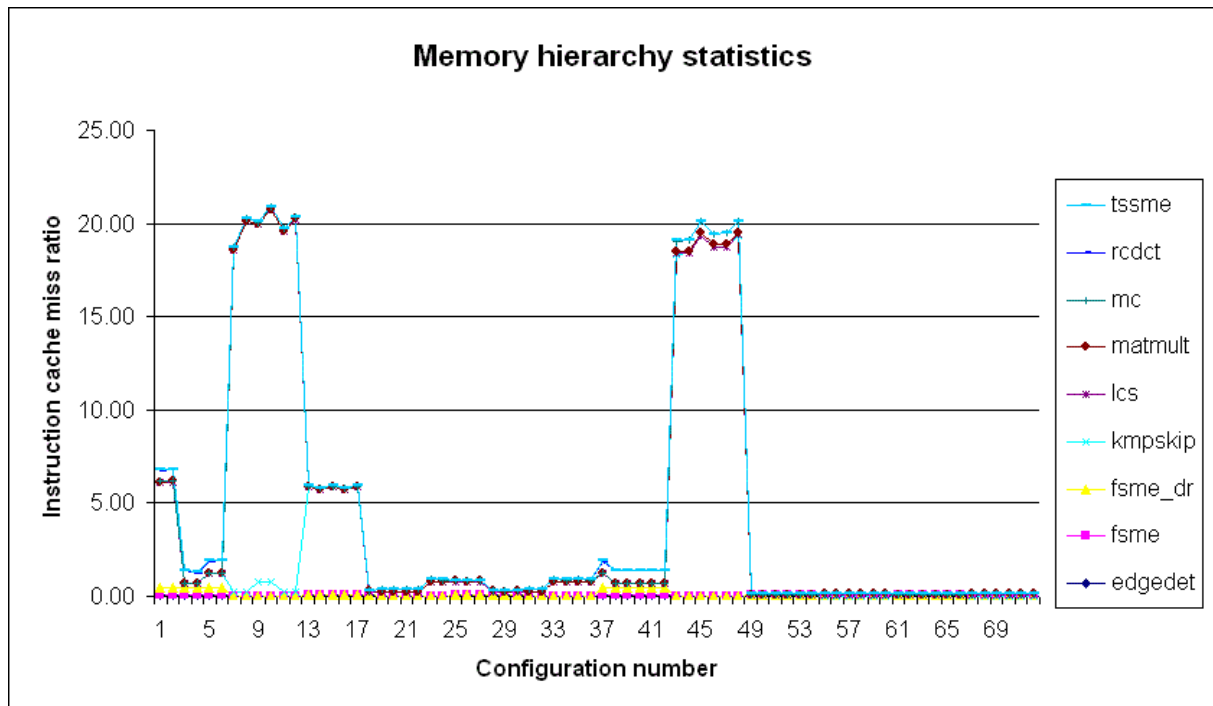
2.4. Διερεύνηση της οργάνωσης της αρχιτεκτονικής μνήμης και καταχωρητών

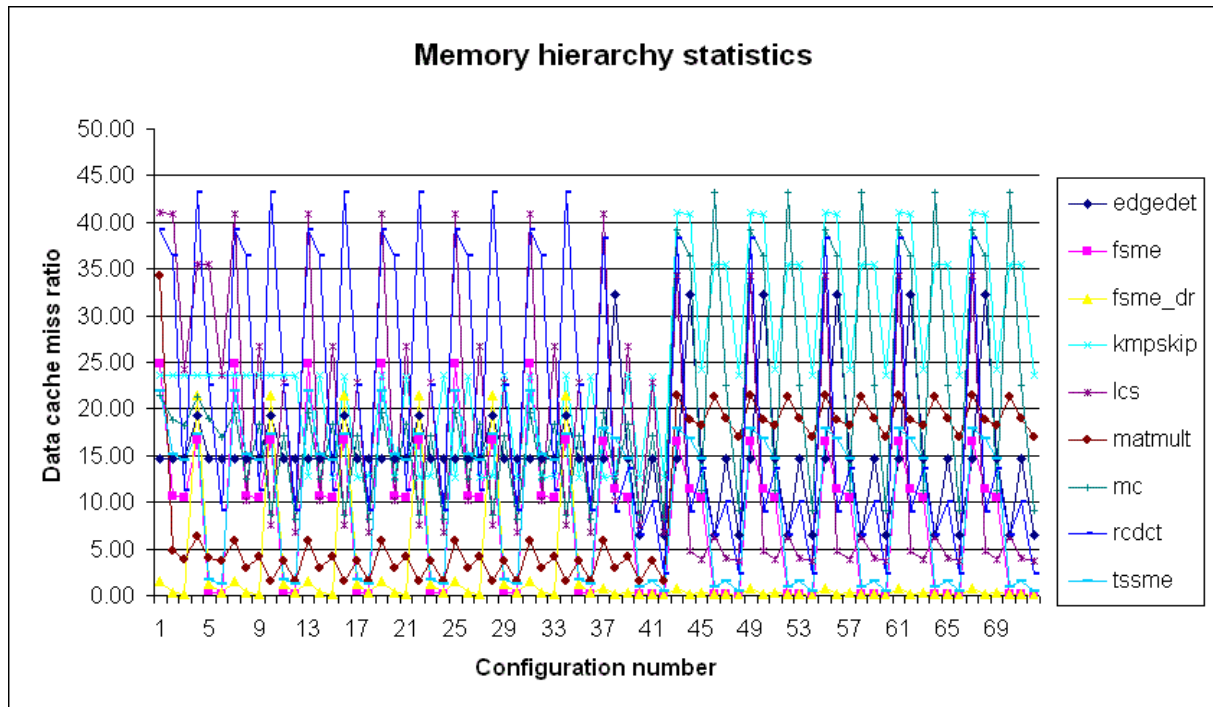
Στο σημείο αυτό θα διερευνηθεί η κατάλληλη αρχιτεκτονική μνήμης προγράμματος και δεδομένων όσον αφορά τις εφαρμογές δοκιμής (τα εφαρμογίδια ZOLCbench της ενότητας 3.χ). Έτσι, θα εξεταστεί η κατάλληλη αρχιτεκτονική κρυφής μνήμης (τοπολογία και παράμετροι όπως το μέγεθος, εύρος σειράς, αλγόριθμος ανανέωσης καταχωρήσεων κ.λ.π.). Η καταλληλότητα της εκάστοτε εξεταζόμενης ρύθμισης (συγκεκριμένες τιμές παραμέτρων αρχιτεκτονικής κρυφής μνήμης) κρίνεται με βάση την αναλογία επιτυχίας (hit ratio) των ανακλήσεων εντολών και αναγνώσεων/εγγραφών δεδομένων από την κρυφή μνήμη. Για το σκοπό αυτό, έγινε σειρά προσομοιώσεων για τον επεξεργαστή DLX (που είναι ένας αντιπροσωπευτικός RISC) ενώ μια αντίστοιχη μελέτη για τον MIPS-I (R3000) και για έναν SPARC-V8 συμβατό μπορεί να βρεθεί στο [Via03]. Αν και στην ArchC [ArchC],[Aze05] υπάρχει η δυνατότητα περιγραφής και ιεραρχημένων δομών κρυφής μνήμης, για τη διερεύνηση θεωρήθηκαν μόνο περιπτώσεις που μπορούν να περιγραφούν σε VHDL στα χρονικά περιθώρια του προγράμματος ή για τις οποίες υπάρχουν διαθέσιμα μοντέλα που μπορούν να προσαρμοστούν κατάλληλα. Έτσι οι τοπολογίες κρυφής μνήμης που

εξετάστηκαν είναι ενός επιπέδου, με ξεχωριστές μνήμες δεδομένων και εντολών. Ο Πίνακας 2.2 δίνει τις παραμέτρους για τις τοπολογίες που εξετάστηκαν, ενώ στο Σχ. 2.1 δίνονται τα αναλυτικά αποτελέσματα.

Πίνακας 2.2: Παράμετροι ιεραρχίας κρυφής μνήμης που εξετάστηκαν.

Instruction cache parameters			
Topology	Lines	Words per line	Write policy
dm	128, 512, 2048	1, 4	Write-through/Write-around
Data cache parameters			
Topology	Lines	Words per line	Write policy
2w, fully, dm	128, 512, 2048	1, 4	Write-back/Write-allocate





(β)

Σχήμα 2.1: % miss ratio για τις ιεραρχίες κρυφής μνήμης. (α) Κρυφή μνήμη εντολών. (β) Κρυφή μνήμη δεδομένων.

Όσον αφορά τις εξειδικευμένες ιεραρχίες μνήμης κατά τη γνωστή μεθοδολογία για την DTSE [Cat98] (η διερεύνηση γίνεται με χρήση μετασχηματισμών επαναχρησιμοποίησης δεδομένων [Lui06]) δεν υπάρχουν διαθέσιμα τα κατάλληλα εργαλεία ανάπτυξης, ήτοι δεν υφίστανται υποδομές μεταγλωττιστή που να υλοποιούν τα βήματα της μεθοδολογίας DTSE και να οργανώνουν ιεραρχικά τις προσπελάσεις σε πολλαπλά επίπεδα στατικών μνημών δεδομένων.

Για την εξέταση της οργάνωσης της αρχιτεκτονικής καταχωρητών είναι αναγκαίο να εκτιμηθούν παράμετροι που προκύπτουν από τη δυναμική ανάλυση των εφαρμογών δοκιμής, όπως:

- ο χρόνος ζωής καταχωρητών, ο οποίος δίνει τα διαστήματα κατά την εκτέλεση μιας εφαρμογής στα οποία μια συγκεκριμένη μεταβλητή πρέπει να είναι αποθηκευμένη σε καταχωρητή.
- ο κορεσμός καταχωρητών (register saturation) [Tou05] ο οποίος αποτελεί χαρακτηριστική ποσότητα για την χειρότερη περίπτωση απαίτησης σε αριθμό διαθέσιμων καταχωρητών για ένα γράφο εξάρτησης δεδομένων (DDG). Τέτοιοι γράφοι χρησιμοποιούνται από τον ερευνητή για την καταγραφή της πληροφορίας σε επίπεδο βασικού μπλοκ.
- Ο αριθμός θυρών ανάγνωσης/εγγραφής για ένα αρχείο καταχωρητών. Ο ερευνητής έχει τη δυνατότητα να κάνει πλήρη ανάλυση για οποιοδήποτε συνδυασμό Ni/No , του αριθμού των ορισμάτων εισόδου και εξόδου κατά τη διερεύνηση για τις απαιτήσεις των ειδικών εντολών.

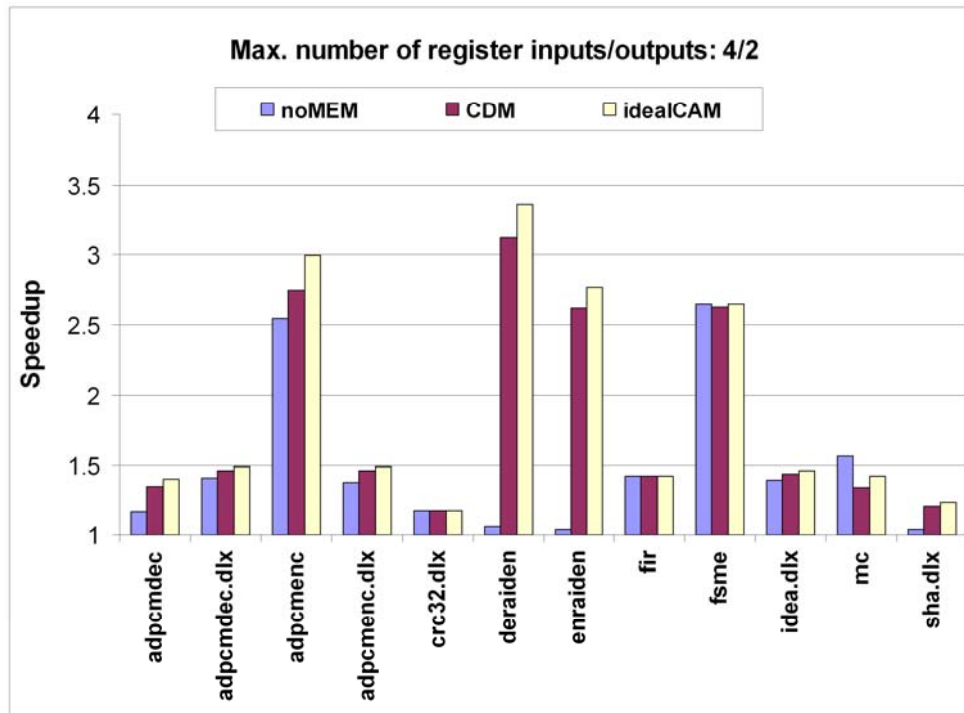
Στα πλαίσια της διερεύνησης για την κατάλληλη αρχιτεκτονική αρχείου καταχωρητών προτείνεται (για τους ASIP που σχεδιάζονται εκ του μηδενός) να εξετάζεται (σε κάθε περίπτωση ξεχωριστά) η χρήση της αρχιτεκτονικής του Εικονικά ή Πραγματικά Απεριόριστου Αρχείου Καταχωρητών (Virtually/Literally Unlimited Register File) το οποίο αναφέρεται από

εδώ και στο εξής και ως VLURF. Μια αντίστοιχη αρχιτεκτονική αρχείου καταχωρητών πρόσφατα προτάθηκε στη σχετική βιβλιογραφία [Sag07]. Ένα στοιχείο αποθήκευσης που ακολουθεί την αρχή VLURF μπορεί να εξυπηρετεί ακόμη το συνδυασμό Ni/No για τις μεγαλύτερες επιτρεπόμενες τιμές των Ni, No. Η απαίτηση σε ορίσματα εισόδου/εξόδου είτε ικανοποιείται λόγω επάρκειας του πραγματικού αριθμού θυρών ανάγνωσης εγγραφής του αρχείου καταχωρητών είτε λόγω εικονικοποίησης (virtualization) της διαδικασίας μεταφοράς ορισμάτων από και προς το αρχείο καταχωρητών. Επίσης, θα εξεταστούν (Φάση 5) οι επιδόσεις (ταχύτητα απόκρισης, επιφάνεια) του VLURF για διαφορετικές τιμές του αριθμού θυρών εισόδου/εξόδου και καταχωρητών.

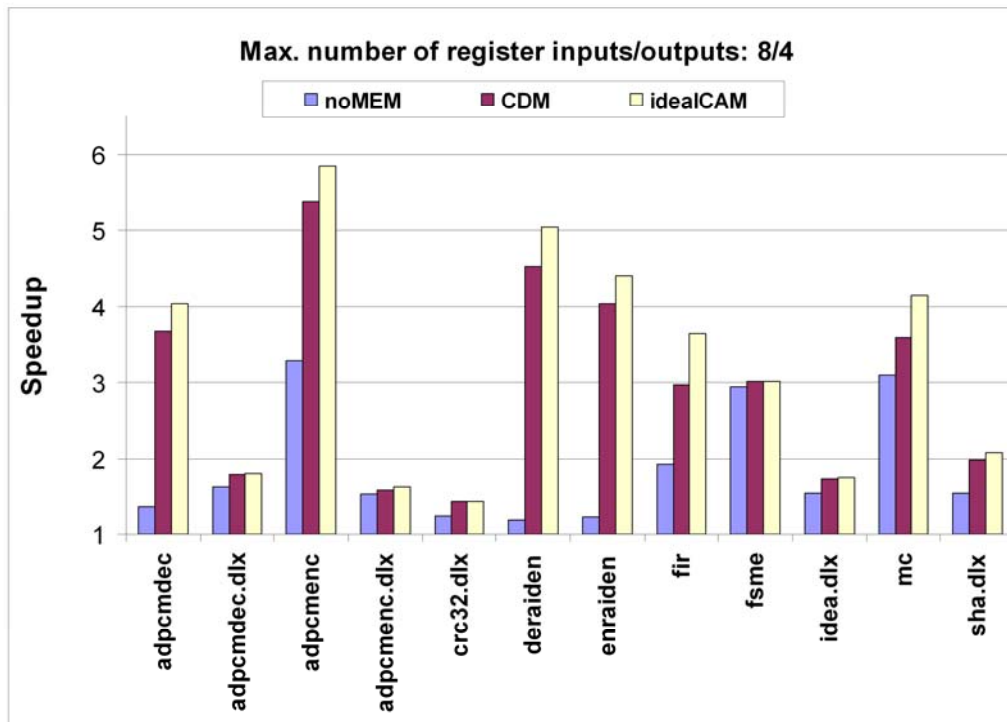
Στη συνέχεια ακολουθεί ένα παράδειγμα διερεύνησης για το μοντέλο πόρων τοπικής αποθήκευσης για την εξυπηρέτηση των ειδικών λειτουργικών μονάδων (ΕΛΜ). Η διερεύνηση χρησιμοποιεί τεχνική ανάλυσης εντολών πολλαπλών-εισόδων/πολλαπλών-εξόδων (MIMO) που θα περιγραφεί διεξοδικά στο παραδοτέο Π4.1 της Φάσης 4.

Η εμβέλεια της χρήσης ειδικών εντολών περιορίζεται από το εύρος bit για την προσπέλαση δεδομένων από την μνήμη δεδομένων και από τα στοιχεία τοπικής αποθήκευσης (αρχείο καταχωρητών) που καθορίζεται από τον αριθμό θυρών εισόδου/εξόδου και την ανάλυση των εξαρτήσεων ανάμεσα σε εντολές φόρτωσης και αποθήκευσης. Σε συγκεκριμένες προσεγγίσεις [Leu06],[Cla05], που ασχολούνται με προκαθορισμένες αρχιτεκτονικές όπως οι MIPS CoExtend και ARM OptimoDE, ο περιορισμός αυτός θέτει έναν δεσμευτικό παράγοντα. Παρ' όλα αυτά, για τον σκοπό της διερεύνησης, κατά τη διάρκεια ανάπτυξης ενός ASIP (π.χ. μιας νέας αρχιτεκτονικής χωρίς δεσμεύσεις συμβατότητας) είναι χρήσιμο να εξετάζονται διαφορετικά μοντέλα συνάφειας μνήμης (memory consistency) όπως αυτή ορίζεται στο [Bis07]. Ακολουθώντας τη σημειολογία που εισήχθη στο [Bis07], η συνάφεια κατάστασης (state consistency) ανάμεσα στις AFUs με τοπική αποθήκευση και την μνήμη δεδομένων (π.χ. μια «πρόχειρη» μνήμη δεδομένων στο ολοκληρωμένο), θεωρούμε δύο διακριτά μοντέλα:

- Τη συναφή μνήμη δεδομένων (consistent data memory), όπου η AFU προσπελαίνει άμεσα την μνήμη δεδομένων και δεν υπάρχει ανάγκη για τοπική αποθήκευση στις AFU. Επίσης κάνουμε τη συντηρητική υπόθεση ότι οι εντολές φόρτωσης και αποθήκευσης πρέπει πάντοτε να είναι σε σειρά (serialized).
- Την ιδανική συναφή μνήμη ΕΛΜ (ideal consistent AFU memory), όπου κάθε φόρτωση/αποθήκευση προς/από την μνήμη δεδομένων μετασχηματίζεται σε μια προσπέλαση στην τοπική μνήμη ΕΛΜ. Θεωρούμε ότι η κατάσταση της μνήμης δεδομένων ανανεώνεται με προσπελάσεις DMA που συμβαίνουν παράλληλα ως προς τις εντολές του ΕΕΣ. Για την διερεύνηση της επίδρασης του μοντέλου μνήμης στην επιτάχυνση των εφαρμογών (παρουσία των ειδικών εντολών), πρώτα παρήχθησαν ειδικές εντολές χωρίς επίτρεψη για την συμπερίληψη τοπικής μνήμης ("noMEM"), έπειτα επιτράπηκε η προσθήκη τοπικής μνήμης και πραγματοποιήθηκαν εκτιμήσεις για το μοντέλο συναφούς μνήμης δεδομένων ("CDM") και εν συνεχεία υποτέθηκε το μοντέλο ιδανικής συναφούς μνήμης AFU ("idealCAM"). Τα αντίστοιχα αποτελέσματα παρουσιάζονται στο Σχ. 2.2 για ενδεικτικούς συνδυασμούς (Ni/No) για επεξεργαστή ΕΕΣ (με βάση τις αρχιτεκτονικές SUIFvm, DLX) με μονή έκδοση εντολής (single-issue).



(α) Ni/No = 4/2



(β) Ni/No = 8/4

Σχήμα 2.2: Επίδραση των προσπελάσεων στη μνήμη δεδομένων πάνω στην επιτάχυνση εφαρμογής που οφείλεται στις ειδικές εντολές. Οι προσπελάσεις στη μνήμη δεδομένων απαιτούν επιβάρυνση ενός κύκλου μηχανής.

Όπως προκύπτει από τα αποτελέσματα, η συμπερίληψη λειτουργιών φόρτωσης και αποθήκευσης στις παραγόμενες ειδικές εντολές έχει σημαντικά θετική επίδραση: η επιτάχυνση εφαρμογών αυξάνεται κατά 15.5 ως 33.3% για τους δοσμένους περιορισμούς

αριθμού ορισμάτων εισόδου/εξόδου. Ιδιαίτερα για την αρχιτεκτονική SUIF_{vm}, οι βελτιώσεις στην επιτάχυνση εφαρμογών είναι γύρω στο 43.4%. Άλλη σημαντική παρατήρηση είναι ότι το μοντέλο συναφούς μνήμης AFU έχει περιορισμένη επίδραση (βελτιώσεις μέχρι 6.3% κατά μέσο όρο και 8.9% για τις εφαρμογές SUIF_{vm} μόνο). Όμως, για μεγαλύτερη επιβάρυνση (σε αριθμό κύκλων) για την προσπέλαση δεδομένων, οι επιταχύνσεις είναι περισσότερο αξιοσημείωτες. Σε ένα άλλο παράδειγμα διερεύνησης, τέθηκε καθυστέρηση ίση με 2 και 5 κύκλους. Εδώ υποτέθηκε ότι η προσπέλαση γίνεται μέσω τοπικού διαύλου (ένας κύκλος διευθυνσιοδότησης ακολουθούμενος είτε από ένα κύκλο δεδομένων μήκους πλήρους λέξης είτε από διαδοχικούς κύκλους προσπέλασης εύρους ενός byte κάθε φορά). Στις περιπτώσεις αυτές υπολογίζονται σημαντικά υψηλότερες επιταχύνσεις. Συγκεκριμένα, η περίπτωση “CDM” αποδίδει καλύτερα από την “noMEM” κατά 34.7% και 49.9% αντίστοιχα, για τις περιπτώσεις των επιβαρύνσεων κατά 2 και 5 κύκλους. Όταν συγκρίνονται μεταξύ τους τα μοντέλα που επιτρέπουν στις λειτουργίες φόρτωσης/αποθήκευσης να είναι μέρος των ειδικών εντολών, οι αντίστοιχες τιμές είναι 7% και 20.9% υπέρ του μοντέλου “idealCAM” για τις δοθείσες επιβαρύνσεις προσπέλασης.

2.5. Καθορισμός των χαρακτηριστικών και της διασυνδεσιμότητας των λειτουργικών μονάδων των ΕΕΣ

Ιδιαίτερα κρίσιμη για την επιτυχία της ροής σχεδιασμού ΕΕΣ συνόλου εντολών είναι η διεξοδική διερεύνηση τόσο της οργάνωσης (εσωτερική δομή και εξυπηρετούμενες λειτουργίες) όσο και των τρόπων επικοινωνίας (διασύνδεση) των λειτουργικών μονάδων των ΕΕΣ. Καταρχήν θεωρούμε (όπως έχει αναφερθεί παραπάνω) ότι η μεθοδολογία στηρίζει δύο διακριτούς τύπους ΕΕΣ:

- ΕΕΣ που ενισχύονται με επέκταση συνόλου εντολών. Πρόκειται για ΕΕΣ με βασικό σύνολο εντολών, ικανό να εξυπηρετεί πλήρως τους υπολογισμούς με ακεραίους σε μια γλώσσα προγραμματισμού υψηλού επιπέδου (όπως η ANSI C). Οι εντολές επέκτασης επιταχύνουν την εκτέλεση συγκεκριμένων τμημάτων κώδικα (των στοχευόμενων εφαρμογών) αλλά π.χ. μπορεί να μην αξιοποιούνται σε λειτουργίες διαχείρισης συστήματος. Για αυτούς τους ΕΕΣ (customizable processors), κρίνεται αναγκαία η ύπαρξη πλήρους εργαλειοθήκης ανάπτυξης εφαρμογών (να συμπεριλαμβάνεται μεταγλωττιστής). Παραδείγματα τέτοιων ΕΕΣ είναι οι Xtensa LX [Tensilica], ARC 710 [ARC], LEON2-CIS [LEON2-CIS] κ.α..
- ΕΕΣ που σχεδιάζονται εκ του μηδενός. Πρόκειται συνήθως για ΕΕΣ με περιορισμένο σύνολο εντολών, και στην βιβλιογραφία αυτού του είδους οι ΕΕΣ (ASIP) αναφέρονται και ως «προγραμματιζόμενοι επιταχυντές» (programmable accelerators). Για τους περισσότερους από αυτούς τους ΕΕΣ δεν αναμένεται η ύπαρξη πλήρους εργαλειοθήκης ανάπτυξης εφαρμογών (αυτό συνήθως ερμηνεύεται σε απουσία μεταγλωττιστή). Παραδείγματα τέτοιων ΕΕΣ είναι ο ARM MOVE coprocessor (που διαθέτει σύνολο 6 εντολών για επιτάχυνση λειτουργιών εκτίμησης κίνησης) [MOVE], ο motion estimation ASIP του [Ger05], ο AMEP [Mom06], ο ICORE [Ric03] κ.α..

Στις παρακάτω ενότητες, καθορίζονται οι παράμετροι (χαρακτηριστικές ιδιότητες) που θα εξετάζονται για την εσωτερική οργάνωση των ειδικών λειτουργικών μονάδων (ενότητες 2.5.1-2.5.2) και ο τρόπος ενσωμάτωσης/διασύνδεσης των μονάδων αυτών σε επεξεργαστικά συστήματα με ΕΕΣ (2.5.3).

2.5.1. Καθορισμός των χαρακτηριστικών ιδιοτήτων για την εσωτερική οργάνωση των ειδικών λειτουργικών μονάδων

Ο καθορισμός των χαρακτηριστικών των ΕΛΜ και ο εν γένει σχεδιασμός τους αποτελεί

σύνθετη διαδικασία. Απαιτεί την εκμετάλλευση των αποτελεσμάτων της ανάλυσης των στοχευόμενων εφαρμογών (με την ροή της Φάσης 2), την εμπειρική αποτίμηση ορισμένων από αυτά, αλλά και την τροφοδότησή τους στην μετέπειτα διαδικασία της γένεσης και επιλογής εντολών (Φάση 4). Κατά τη γένεση και επιλογή εντολών, καθορίζονται αυστηρώς οι εξής λεπτομέρειες για την εσωτερική οργάνωση των λειτουργικών μονάδων, οι οποίες είναι και οι χαρακτηριστικές ιδιότητες των ΕΛΜ όσον αφορά την εσωτερική τους οργάνωση:

- Οι λειτουργίες που επιτελούνται (functionalities).
- Η εσωτερική μικροαρχιτεκτονική οργάνωση όσον αφορά τον χρονισμό των επιτελούμενων μικρολειτουργιών: συνδυαστική (combinational), με διοχέτευση (pipelined), πολλαπλών κύκλων/επαναληπτική (multi-cycle).
- Αριθμός και οργάνωση (διευθυνσιοδότηση, προσπέλαση) τοπικών πόρων αποθήκευσης.
- Ο αριθμός ορισμάτων εισόδου/εξόδου και το εύρος bit για κάθε μία είσοδο και έξοδο. Το πρώτο γίνεται αυτόματα ενώ το δεύτερο χειρωνακτικά από τον σχεδιαστή, εφόσον δεν είναι διαθέσιμο πέρασμα μεταγλωττιστή ανάλυσης εύρους ψηφίου (bitwidth analysis).
- Το διαθέσιμο εύρος ζώνης για την ταυτόχρονη ανάγνωση/εγγραφή ορισμάτων εισόδου/εξόδου, αντίστοιχα.
- Την δρομολόγηση των μικρολειτουργιών ανάγνωσης/εγγραφής ορισμάτων εισόδου/εξόδου, αντίστοιχα.
- Ο διαμοιρασμός πόρων (resource sharing) ανάμεσα σε δύο ή παραπάνω εντολές.
- Ο χρονισμός για την ανάγνωση όλων των ορισμάτων εισόδου, τον συνολικό χρόνο (κύκλοι μηχανής) για την επεξεργασία των δεδομένων (από την ανάγνωση της πρώτης εισόδου μέχρι το πέρας της επεξεργασίας: cadency, από την ανάγνωση της τελευταίας εισόδου: latency), και τον ελάχιστο αριθμό κύκλων για την πρόσληψη νέων ορισμάτων/ παραγωγή ενός νέου αποτελέσματος (throughput).

Ο υπολογισμός των τιμών για τα χαρακτηριστικά των λειτουργικών μονάδων γίνεται με την αποτίμηση των αποτελεσμάτων της στατικής και δυναμικής ανάλυσης των στοχευόμενων εφαρμογών, (χρησιμοποιούμενοι τύποι δεδομένων, συχνότητα εμφάνισης εντολών/λειτουργιών, εγγενής παραλληλία της εφαρμογής κ.α.).

Σε ορισμένες περιπτώσεις θα μπορούσαν να χρησιμοποιηθούν στο SoC σύστημα μη-προγραμματιζόμενες λειτουργικές μονάδες (ASIC) που έχουν σχεδιαστεί χειρωνακτικά. Τέτοιες περιπτώσεις είναι: μονάδες για μετασχηματισμούς πεδίου όπως FFT, DCT και χρωματικού πεδίου (π.χ. YUV-to-RGB), υπολογισμών γωνιών και μετασχηματισμών συντεταγμένων κατά CORDIC, υπερβατικών αριθμητικών συναρτήσεων, και γενικά μονάδες για λειτουργίες που μπορούν να επιτελεστούν με πλήρη αυτονομία ως προς το διάυλο ελέγχου του ΕΕΣ. Επίσης, ένα άλλο κοινό χαρακτηριστικό τους είναι ότι υλοποιούν λειτουργικότητες που δεν μεταβάλλονται από εφαρμογή σε εφαρμογή. Για τέτοιους επιταχυντές συναρτήσεων (accelerators) υπάρχει σημαντικός αριθμός δωρεάν διαθέσιμων υλοποιήσεων [Opencores].

2.5.2. Ένα πλήρες παράδειγμα: Ανάλυση του αλγορίθμου συμπίεσης μορφής κατά MPEG-4 και σχεδιασμός της μονάδας για την λειτουργία «ανεύρεσης διανυσμάτων κίνησης»

Για το παράδειγμα αυτό, η εφαρμογή του ενδιαφέροντος είναι ένας context-based κωδικοποιητής μορφής για το πρότυπο συμπίεσης video MPEG-4 [MPEG4-VVM18],[MPEG4senc]. Ο κωδικοποιητής επιτρέπει τις απωλεστικές αποφάσεις στην κωδικοποίηση της πληροφορίας μορφής, και ελέγχεται από δύο παραμέτρους: την

(*motion_th*) που σχετίζεται με το διάνυσμα κίνησης και την (*alpha_th*) που εκφράζει τον βαθμό των απωλειών. Η παράμετρος *motion_th* επιτρέπει τον έλεγχο της ακρίβειας των διανυσμάτων κίνησης: *motion_th*=0 αντιστοιχεί στη μέγιστη δυνατή ακρίβεια, ενώ ένα μη-μηδενικό *motion_th* υπονοεί ότι για μακρομπλόκ εικόνας με χαμηλή κίνηση (low-motion), κωδικοποιείται ένα διάνυσμα πρόβλεψης για το μέτρο της κίνησης και δεν εκτελείται η πλήρης διαδικασία εκτίμησης κίνησης. Ο παράγοντας κατωφλίου για τα μπλοκ *alpha_th* επιτρέπει την απωλεστική συμπίεση: κάθε μπλοκ ορίζεται ως όλα-0 ή όλα-255 (για τις τιμές των εικονοστοιχείων του) ανάλογα με τον συντελεστή επιτρεπόμενης ποιότητας (Accepted Quality – ACQ) και συγκρίνεται με την τιμή κατωφλίου. Στα πειράματα του [Kan05b] κάθε πλαίσιο-κλειδί σε μια ακολουθία P-VOP (τυπικά αποτελούμενη από 10 πλαίσια εικόνας) κωδικοποιείται σε *intra mode* ενώ τα εναπομείναντα πλαίσια σε *inter mode*.

Για την εφαρμογή “mpeg4_senc” καταγράφηκε το δυναμικό προφίλ για τις 16 περιπτώσεις εκτέλεσης που προσδιορίζονται από το παραμετρικό πεδίο: {*alpha_th*, *motion_th*}={0, 32, 64, 256}. Έτσι, εξήχθει ότι τα πιο κρίσιμα (άνω του 70% των συνολικών εκτιμώμενων κύκλων εκτέλεσης) είναι τα BB #40 and #41 της συνάρτησης *find_vects*, που περιλαμβάνει τον καθ’ αυτό κώδικα για τον υπολογισμό του κριτηρίου SAD. Ο Πίνακας 2.3 συνοψίζει τα στατιστικά για τις αυτόματα γεννώμενες ειδικές εντολές και τους σχετικούς κύκλους δρομολόγησης για αυτές για 3 διαφορετικά σενάρια περιορισμών και συγκεκριμένο, κάθε φορά, μέγιστο αριθμό ορισμάτων εισόδου (στήλη “#inputs”). Στις στήλες “#MaxMISO” και “MaxMISO size”, δίνονται αντίστοιχα ο αριθμός των στατικών εμφανίσεων και το μέγεθος (ως αριθμός στοιχειωδών λειτουργιών) των ειδικών εντολών. Από τα αποτελέσματα του Πίνακα 2.3 συμπεραίνουμε ότι ο περιορισμός στον αριθμό εισόδων ανά ειδική εντολή διαδραματίζει σημαντικό ρόλο στο μέγεθος της παραγόμενης εντολής τύπου MISO. Οι επιτυγχανόμενες επιταχύνσεις κυμαίνονται από 1.07 ως 3.50, με την τελευταία περίπτωση (απεριόριστες εισοδοί) να είναι δυνατό να υλοποιηθεί με τη χρήση καταχωρητών κατάστασης (state registers) που μειώνουν σημαντικά την απαίτηση σε ορίσματα εισόδου από το αρχείο καταχωρητών του ΕΕΣ.

Από το σημείο αυτό, σε κάθε βασικό μπλοκ ανατίθεται μοναδικό όνομα με την διαμόρφωση: <procedure_name>.<basic_block_number>. Το Σχήμα 2.3.α δείχνει τον DDG για την MISO μέγιστης επιτάχυνσης που αναγνωρίστηκε για το βασικό μπλοκ *find_vects*.40 κάτω από τους εξής περιορισμούς κόμβων: {Type-A, Type-B} = {*str*, *cal*}. Οι εκτιμήσεις επιφάνειας υλικού και χρονικής καθυστέρησης (area, delay) του Σχήματος 2.3 έχουν κανονικοποιηθεί ως προς τις τιμές ενός πολλαπλασιαστή μονού κύκλου 32×32-bit που επιστρέφει αποτέλεσμα εύρους 64-bit (μη συντετημημένο). Παρατηρώντας τον γράφο του Σχ. 2.3.α εξάγονται σημαντικά συμπεράσματα:

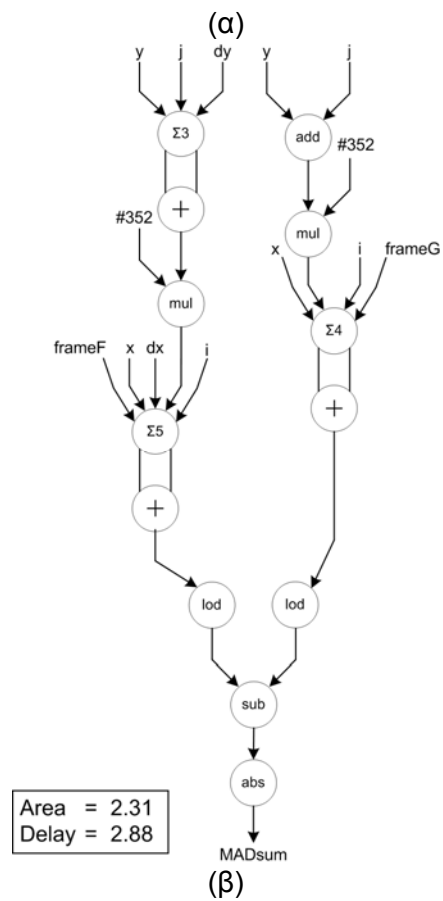
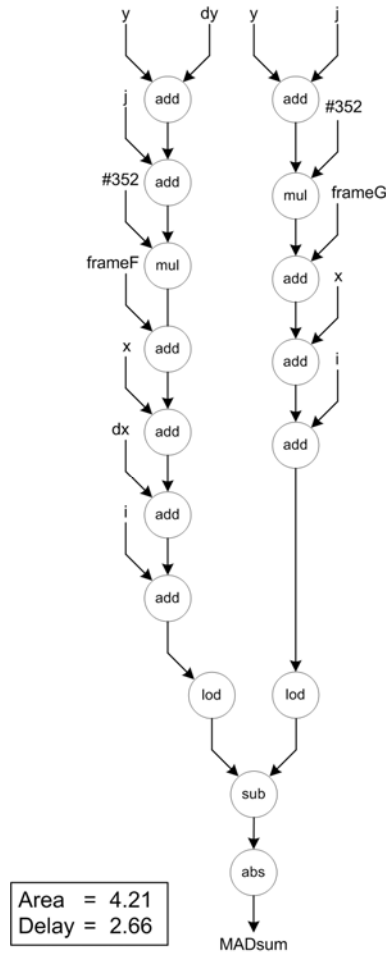
- 1) Υπάρχουν τρεις διακριτές εμφανίσεις άθροισης πολλαπλών ορισμάτων (multi-operand addition) με 3-, 4- και 5- ορίσματα. Θα μπορούσαν να εξαχθούν και εκτενέστερες δομές άθροισης πολλαπλών ορισμάτων με την συμπερίληψη και των πολλαπλασιασμών-με-σταθερά που εμφανίζονται στην ειδική εντολή (σχολιασμός στο σημείο 2). Οι αθροιστές πολλαπλών ορισμάτων έχουν χαρακτηριστεί από μία δημόσια VHDL βιβλιοθήκη [Beu04] για τις εκτιμήσεις επιδόσεων. Το Σχήμα 2.3.β δείχνει την MISO (η οποία είναι και MaxMISO) με όλες τις εμφανίσεις άθροισης πολλαπλών ορισμάτων να έχουν αντικατασταθεί με συμπιεστές αποθήκευσης κρατούμενου n:2.
- 2) Δεν υπάρχει καμία εμφάνιση πολλαπλασιασμού μεταξύ μεταβλητών αλλά μόνο πολλαπλασιασμοί με σταθερά. Η σταθερά είναι το εύρος του πλαισίου εικόνας (#352 για την ακολουθία δοκιμής “stefan”). Τυπικά εύρη πλαισίου για ρεύμα video είναι τα: 120 (SQCIF), 176 (QCIF), 352 (CIF), και 704 (4CIF). Στο Σχ. 2.3.γ απεικονίζεται ο γράφος ροής δεδομένων για υλοποίηση λειτουργικής μονάδας (για την ειδική εντολή) που ελαττώνει περαιτέρω την συνδυαστική καθυστέρηση και επιφάνεια σε σχέση με το Σχ. 2.3.β αξιοποιώντας πολλαπλασιαστές-με-σταθερά. Στην περίπτωση που πρέπει να υποστηρίζεται σύνολο σταθερών, μπορούν αντίστοιχα να χρησιμοποιηθούν πολλαπλασιαστών πολλαπλών σταθερών της βιβλιογραφίας.

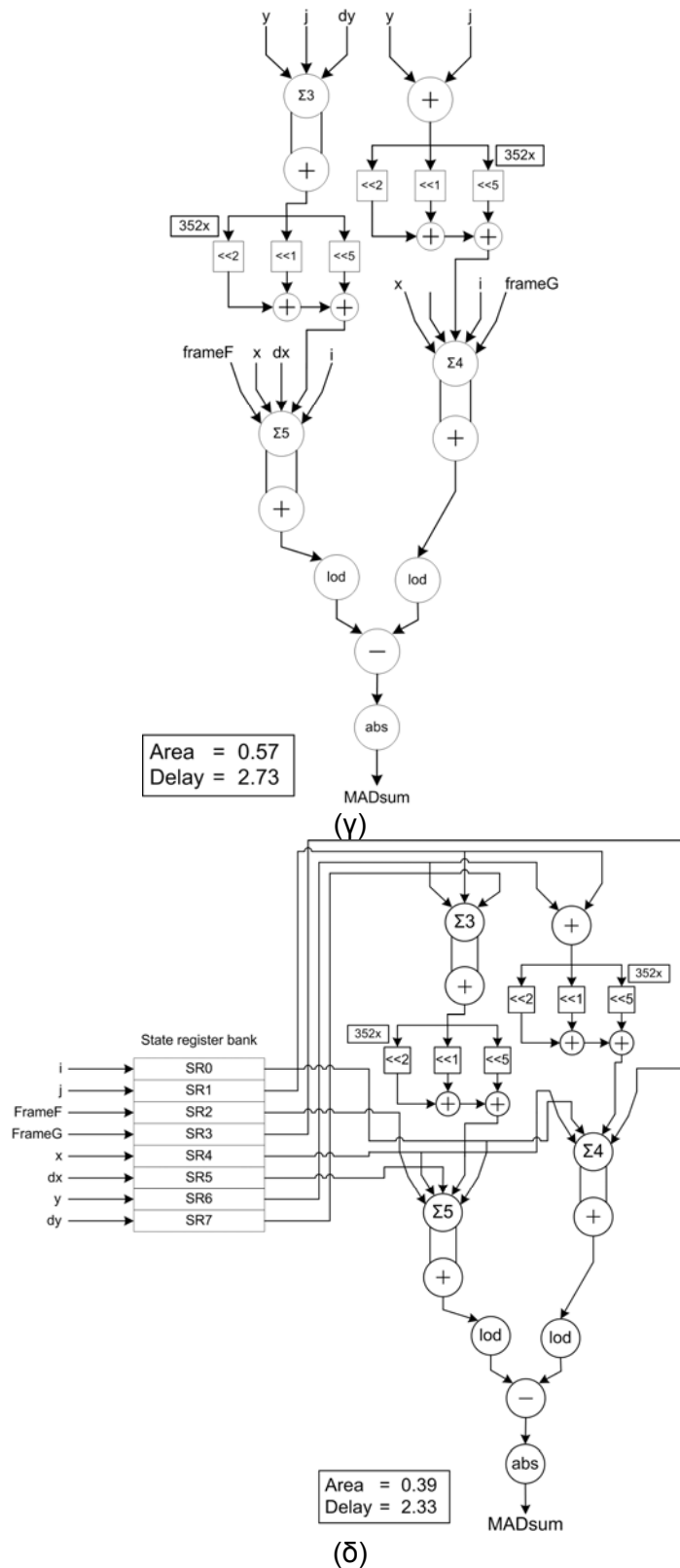
- 3) Οι δύο εντολές φόρτωσης (*lod*) μπορούν να εκτελεστούν ταυτόχρονα μόνο σε περίπτωση που η μνήμη δεδομένων διαθέτει (παραπάνω από μία) θύρες με επίτρεψη παράλληλης προσπέλασης.
- 4) Ο εγγενής ILP του DDG του Σχ. 2.3.α είναι αρκετά χαμηλός (1.6) ενώ ILP του 2.18 σημειώνεται με προσομοίωση για επεξεργαστή του συνόλου εντολών PISA (MIPS-I) με 4 θυρίδες εκτέλεσης (4-way) με τον SimpleScalar 3.0d [Bur97],[Aus02] για την όλη εφαρμογή. Ο μεταγλωττιστής είναι ο *gcc-2.7.2.3* στοχευμένος στην αρχιτεκτονική SimpleScalar PISA με ενεργοποιημένες βελτιστοποιήσεις επιπέδου -O3. Μπορεί εδώ να λεχθεί ότι η ύπαρξη χαμηλού ILP ευνοεί την εκμετάλλευση των αλυσιδωτών (chained) λειτουργιών παρά την παράλληλη εκτέλεση τους από VLIW εντολές. Σε αντίθεση με τις VLIW αρχιτεκτονικές που εκδίδουν απλές ανεξάρτητες λειτουργίες με την κάθε μια να δεσμεύει μία θυρίδα εκτέλεσης ανά βήμα ελέγχου, οι επεξεργαστές με εντολές επέκτασης που υλοποιούν αλυσιδωτές λειτουργίες, αξιοποιούν εξαρτώμενες, μεταξύ τους, εντολές. Σημειώνεται ότι εφαρμογές πολυμέσων όπως οι MPEG-4 visual (version 1 and 2), και H.264/AVC έχουν χαμηλό εγγενές low ILP (περίπου 2 για τις λειτουργίες κωδικοποίησης και αποκωδικοποίησης) [Fri04] το οποίο δεν είναι εκμεταλλεύσιμο από τυπικούς VLIW επεξεργαστές.

Πίνακας 2.3: Αναλυτικά στατιστικά για τις παραχθείσες ειδικές εντολές όπως εξήχθησαν από τα κρίσιμα βασικά μπλοκ του κωδικοποιητή μορφής.

Basic block	#inputs	Node constraints		#MaxMISO	MaxMISO size	Est. relative %cycles	Speedup
		Type A	Type B				
find_vects.40	2	str	cal	5	2	73.68	1.36
find_vects.41	2	str	cal	2	2		
find_vects.40	2	str, lod	cal	5	2	73.68	1.36
find_vects.41	2	str, lod	cal	2	2		
find_vects.40	2	str, lod	cti	5	2	86.61	1.15
find_vects.41	2	str, lod	cti	2	2		
find_vects.40	4	str	cal	6	2.33	47.37	2.11
find_vects.41	4	str	cal	2	3		
find_vects.40	4	str, lod	cal	6	2.67	47.37	2.11
find_vects.41	4	str, lod	cal	2	3		
find_vects.40	4	str, lod	cti	6	2.67	65.22	1.53
find_vects.41	4	str, lod	cti	2	2.5		
find_vects.40	∞	str	cal	1	16	28.57	3.50
find_vects.41	∞	str	cal	1	6		
find_vects.40	∞	str, lod	cal	3	4.67	43.61	2.29
find_vects.41	∞	str, lod	cal	2	3		
find_vects.40	∞	str, lod	cti	3	4.67	48.78	2.05
find_vects.41	∞	str, lod	cti	2	3		

Το τελικό υλικό για την AFU (Σχήμα 2.3.δ) της ειδικής εντολής του Σχ. 2.3.α μπορεί να βελτιωθεί ακόμη περισσότερο με την χειροκίνητη προσθήκη καταχωρητών κατάστασης, όπως υποκινείται από τα αποτελέσματα της ανάλυσης ζωής καταχωρητών. Οι μέγιστες επιδόσεις μπορούν να προσεγγιστούν με την προσθήκη 8 καταχωρητών κατάστασης (Σχ. 2.3.γ). Επίσης το εύρος bit των τελεστών επιλέχθηκε να είναι τα 8-bit για τους *sub* και *abs* και για τους υπόλοιπους τα 16-bit. Συνολικά, ο συνδυασμός αυτόματων και μη βελτιστοποιήσεων επιφέρει μείωση κατά 85% και 40% σε επιφάνεια και χρονική απόκριση, αντίστοιχα.





Σχήμα 2.3: Βήματα προς την βελτιστοποίηση της AFU για την ειδική εντολή που αναγνωρίστηκε στο βασικό μπλοκ `find_vects.40`. (α) Αρχικό DFG. (β) Εμφανίσεις άθροισης πολλαπλών ορισμάτων στο DFG. (γ) Εκμετάλλευση των πολλαπλασιασμών-με-σταθερά. (δ) Εκμετάλλευση καταχωρητών κατάστασης για τοπική αποθήκευση ορισμάτων εισόδου/εξόδου.

2.5.3. Καθορισμός της διασυνδεσιμότητας των ειδικών λειτουργικών μονάδων στους ΕΕΣ

Κεντρικό ρόλο στη μεθοδολογία σχεδιασμού ΕΕΣ κατέχει η διαδικασία εξαγωγής των κατάλληλων αρχιτεκτονικών επεκτάσεων υλικού για κάθε στοχευόμενη εφαρμογή (application-specific hardware extensions ή ASHEs). Μετά την επιτυχή εξαγωγή/ταυτοποίηση των επεκτάσεων αυτών (κάνοντας χρήση τεχνικών γένεσης και επιλογής ειδικών εντολών – Φάση 4), το έτερο ζητούμενο είναι ο κατάλληλος τρόπος ενσωμάτωσής τους στους σχεδιαζόμενους ΕΕΣ (customizable processors ή ASIP). Η ενσωμάτωση των ΕΛΜ που εξυπηρετούν τις αρχιτεκτονικές επεκτάσεις πρέπει να γίνεται με συγκεκριμένους τρόπους. Αυτό εξασφαλίζεται με τον αυστηρό καθορισμό των επιτρεπόμενων τρόπων (πρωτοκόλλων) επικοινωνίας των ΕΛΜ με την υπόλοιπη λογική του ΕΕΣ.

Για τον σκοπό αυτό, στο σημείο αυτό διακρίνουμε τρεις διαφορετικούς τρόπους σύζευξης (coupling) των ΕΛΜ ως προς τον ΕΕΣ (η αναγκαιότητα της μη δέσμευσης σε ένα μόνο τρόπο σύζευξης έχει απασχολήσει την βιβλιογραφία [Sir07]). Οι κύριες διαφορές των τρόπων αυτών έγκεινται:

- στην άμεση ή έμμεση διεπαφή τους ως προς τον ΕΕΣ
- στην υλοποίηση της λογικής ελέγχου τους
- στην εξάρτησή τους από ιδιότητες του ΕΕΣ όπως η συχνότητα χρονισμού (αν επιτρέπεται να χρονίζονται σε διαφορετική συχνότητα από αυτόν)
- η δρομολόγηση των λειτουργιών που επιτελούν ως προς τις λειτουργίες του ΕΕΣ

Αναλυτικά οι αποδεκτοί τρόποι σύζευξης/διασύνδεσης των ΕΛΜ σε ένα SoC σύστημα με ΕΕΣ της μεθοδολογίας είναι οι εξής:

1) ΕΛΜ σε στενή σύζευξη (tightly-coupled ASHEs)

Οι ΕΛΜ της κατηγορίας αυτής ενσωματώνονται εντός της αρχιτεκτονικής διοχέτευσης του ΕΕΣ, και ο έλεγχός τους δεν διαφέρει από αυτόν των ενδεχόμενων βασικών λειτουργικών μονάδων (ΒΛΜ) που εκτελούνται στον ΕΕΣ. Για την συστηματοποίηση της διαδικασίας ενσωμάτωσής τους υιοθετείται ρυθμιζόμενο σύνολο σημάτων διεπαφής εμπνευσμένο από την τακτική ενσωμάτωσης των ειδικών εντολών στους επιτυχημένους soft-core επεξεργαστές Nios-II [Nios-II],[Nios-II-CI]. Η προσέγγιση του Nios-II είναι καθορισμένη με σαφήνεια ενώ έχει χρησιμοποιηθεί με μεγάλη επιτυχία τα τελευταία χρόνια τόσο σε εμπορικό όσο και σε ακαδημαϊκό επίπεδο [Nios-II-DCB]. Οι βασικές διαφορές ως προς την προσέγγιση του Nios-II είναι ότι:

- ο αριθμός ορισμάτων εισόδου/εξόδου δεν περιορίζεται εκ προοιμίου σε 2 και 1, αντίστοιχα
- επιτρέπεται η άμεση αλλά και υπονοούμενη (implicit) διευθυνσιοδότηση τοπικών καταχωρητών
- η κωδικοποίηση της διαμόρφωσης των ειδικών εντολών εξαρτάται από τη διαθεσιμότητα στον χάρτη διαμορφώσεων του ΕΕΣ

Επίσης, οι ΕΛΜ της κατηγορίας ενδεχόμενα μπορούν να δρομολογηθούν σε ταυτόχρονα εξυπηρετούμενες χρονοθυρίδες ως προς τους ΒΛΜ του ΕΕΣ. Ένα ενδεικτικό σχήμα μιας τέτοιας διεπαφής παρουσιάζεται στο Σχ. 2.4.α.

2) ΕΛΜ σε σύζευξη τοπικού συνεπεξεργαστή (ASHEs with local coprocessor coupling)

Οι ΕΛΜ της κατηγορίας αυτής δεν ενσωματώνονται εντός της αρχιτεκτονικής

διοχέτευσης του ΕΕΣ, αλλά τους διατίθεται τοπική διασύνδεση με τον ΕΕΣ η οποία ακολουθεί εξειδικευμένο πρωτόκολλο. Τέτοια εξειδικευμένα πρωτόκολλα είναι το CPI (Coprocessor Interface) για την σύζευξη συνεπεξεργαστών (όπως η μονάδα κινητής υποδιαστολής [GRFPU], ο κρυπτοεπεξεργαστής της αρχιτεκτονικής Thumbrod [Schaum03]), αλλά και γενικευμένα πρωτόκολλα τοπικής διεπαφής όπως αυτά που περιγράφονται στα [Sin03],[Kod03] με τον επεξεργαστή LEON [Gaisler]. Ευρέως χρησιμοποιούμενος τρόπος διασύνδεσης της κατηγορίας αυτής περιλαμβάνει μονοκατευθυντικούς συνδέσμους FSL (Fast Simplex Link) [FSL] για την τοπική διασύνδεση συνεπεξεργαστών (όπως π.χ. μιας μονάδα DCT ή FFT) με επεξεργαστή MicroBlaze. Για τους ΕΕΣ της μεθοδολογίας θα καθοριστεί διεπαφή για την τοπική σύζευξη ενός ή παραπάνω συνεπεξεργαστών, η οποία συνδυάζει στοιχεία από τις προαναφερθείσες περιπτώσεις που έχουν βάση το CPI του LEON αλλά και από κλασσικά πρωτόκολλα διασύνδεσης σημείο-σε-σημείο και σημείο-σε-πολλαπλά-σημεία.

Για το τελευταίο, η διασύνδεση SimpCon [SimpCon] διαθέτει σχεδόν όλα τα κατάλληλα χαρακτηριστικά:

- Για τον master (π.χ. το στάδιο αποκωδικοποίησης εντολής του ΕΕΣ) υπάρχει απαίτηση για επικοινωνία με τον slave μόνο για ένα κύκλο. Αυτό σημαίνει ότι έπειτα είναι ελεύθερος να συνεχίσει με τις εσωτερικές του λειτουργίες. Έτσι, αυτή η προδιαγραφή θα τροποποιηθεί ώστε ο master να συνεχίζει με την εκτέλεση εντολών σε ΒΛΜ ή ΕΛΜ σύζευξης τύπου 1, εφόσον αυτές έχουν δρομολογηθεί.
- Ο master μπορεί να καθυστερήσει την αποδοχή των αποτελεσμάτων από τον τοπικό συνεπεξεργαστή για όσο είναι απαραίτητο (ώστε π.χ. να ολοκληρώσει τοπικές λειτουργίες που έχουν ενδεχομένως μη ικανοποιήσιμες εξαρτήσεις ως προς τα αποτελέσματα του συνεπεξεργαστή).
- Οι slaves μπορούν να έχουν εσωτερική αρχιτεκτονική με στάδια διοχέτευσης.
- Επιτρέπεται η πρόωρη σηματοδότηση ολοκλήρωσης λειτουργιών στους τοπικούς συνεπεξεργαστές (slaves) κάτι που δίνει τη δυνατότητα για έγκαιρη προετοιμασία της απόκρισης του master.

Ένα παράδειγμα τέτοιας διεπαφής δίνεται στο Σχήμα 2.4.β. Η σύζευξη τύπου-2 είναι «στενή», δηλ. οι διαδικασίες ελέγχου (ή το βασικό μέρος τους) επιτελούνται από τη μονάδα ελέγχου του ΕΕΣ.

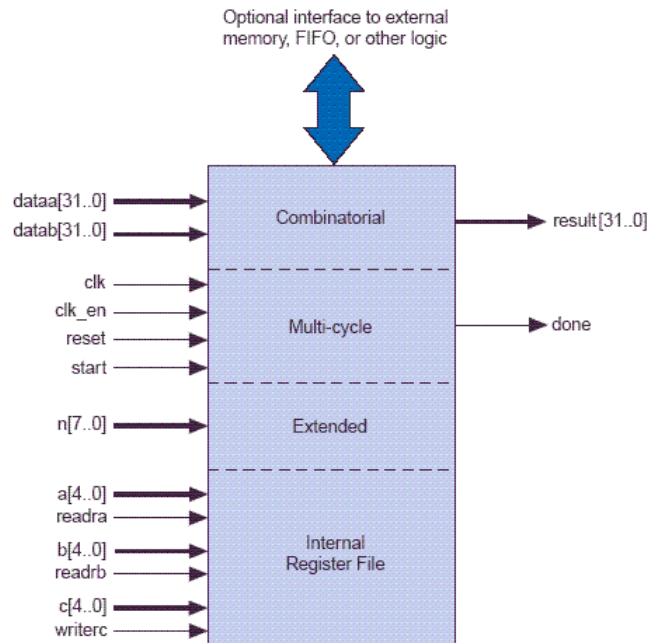
3) ΕΛΜ σε χαλαρή σύζευξη μέσω διαύλου (ASHEs with bus coupling)

Οι ΕΛΜ της κατηγορίας αυτής διασυνδέονται μέσω του διαύλου του υποσυστήματος του επεξεργαστή ΕΕΣ (κατά τα πρότυπα CoreConnect PLB και το επιλεγμένο AMBA 2.0 AHB). Σε αυτές τις περιπτώσεις ανήκουν οι (προγραμματιζόμενοι ή μη) επιταχυντές υλικού (hardware accelerators) όπως μονάδες κρυπτογράφησης [OC-DES], CORDIC [OC-CORDIC], CRC [UCRC] κ.α..

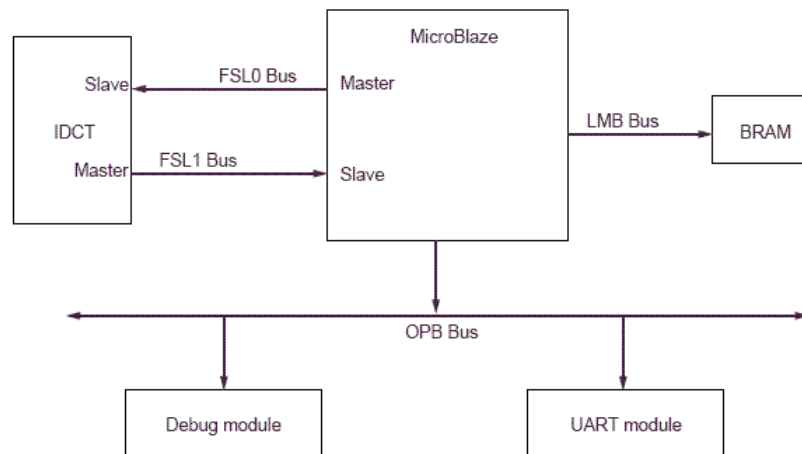
Συνήθως για τη διασύνδεση ΕΛΜ μέσω διαύλου συντρέχουν οι παρακάτω λόγοι:

- Ο ΕΛΜ επιτελεί μια σειρά από εύκολα αυτονομήσιμες (από τον ΕΕΣ) λειτουργίες που μπορούν να ελεγχθούν από σχετικά απλά FSM.
- Ο θεωρούμενος ΕΛΜ χρειάζεται μεγάλο εύρος ζώνης και συχνές προσπελάσεις από/προς την μνήμη δεδομένων.
- Ο ΕΛΜ υλοποιεί έναν αλγόριθμο ο οποίος χρησιμοποιείται χωρίς αλλαγές (είναι τυποποιημένο μπλοκ) σε διαφορετικές εφαρμογές.

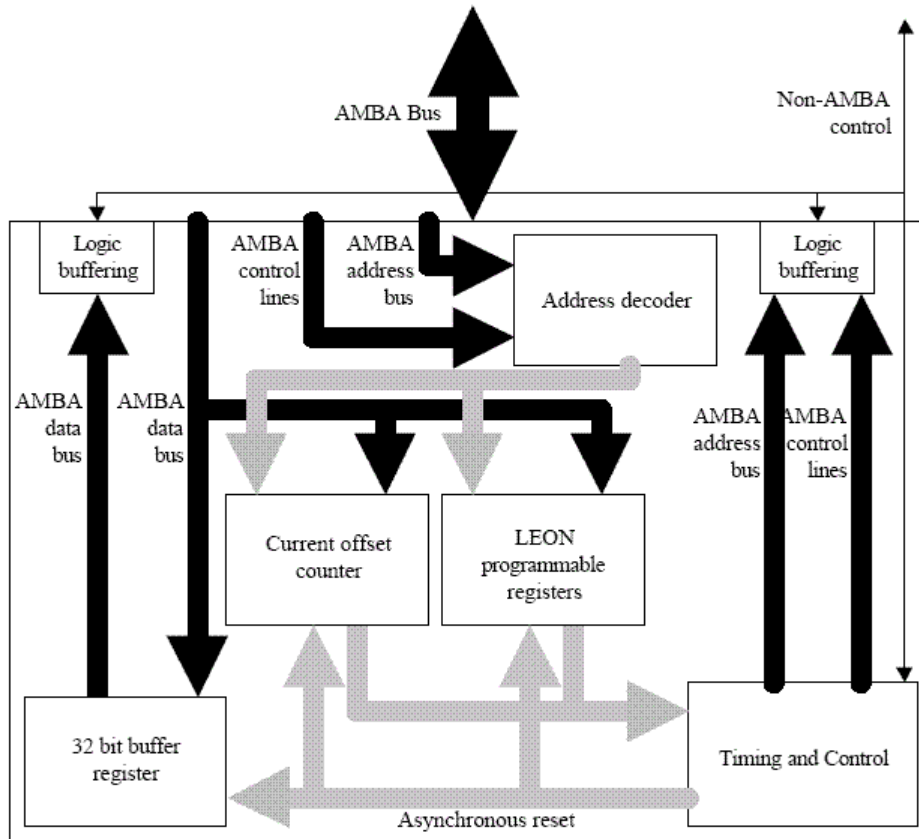
Η διασύνδεση κατά τον τύπο-3 απεικονίζεται στο Σχ. 2.4.γ.



(α) ΕΛΜ σε στενή σύζευξη: παράδειγμα διεπαφής για τις εντολές επέκτασης στον Nios-II.



(β) ΕΛΜ σε σύζευξη τοπικού συνεπεξεργαστή: παράδειγμα διεπαφής για τοπικό συνεπεξεργαστή FFT του MicroBlaze, διασύνδεση μέσω FSL (χρήση FIFO).



(γ) ΕΛΜ σε χαλαρή σύζευξη μέσω διαύλου: παράδειγμα μονάδας DMA για επικοινωνία μέσω AMBA 2.0 AHB με επεξεργαστή LEON-2 SPARC-V8 συμβατό.

Σχήμα 2.4: Αποδεκτοί τρόποι για την σύζευξη ειδικών λειτουργικών μονάδων (ΕΛΜ) στο σύστημα του ΕΕΣ.

3. Η ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΕΛΕΓΧΟΥ ΒΡΟΧΩΝ ZOLC

Οι λειτουργίες εκτέλεσης βρόχων επιβάλλουν σημαντικούς περιορισμούς στην επίτευξη υψηλότερων υπολογιστικών επιδόσεων για τις ενσωματωμένες εφαρμογές. Για την αντιμετώπιση του ζητήματος αυτού, προτάθηκε από τον ερευνητή Α.Π.Θ. μια καινοτόμα αρχιτεκτονική ελέγχου βρόχων μηδενικής επιβάρυνσης (Zero-Overhead Loop Controller – ZOLC) που υποστηρίζει αυθαίρετες δομές βρόχων με κόμβους πολλαπλών-εισόδων και πολλαπλών-εξόδων, η οποία βελτιώνει σημαντικά τις επιδόσεις ενσωματωμένων RISC επεξεργαστών. Προκειμένου την στοχεύσει κώδικα συμβολομεταφραστή ώστε να λαμβάνει υπόψη την αρχή ZOLC, αναπτύχθηκε φορμαλισμός γράφου για την αναπαράσταση των δομών βρόχου που απαντώνται στα προγράμματα εφαρμογών. Επίσης, παρουσιάζεται σε λεπτομέρεια, η μεταφερτή (portable) περιγραφή ενός στοιχείου ZOLC σε μορφή ψευδοκώδικα, η οποία και μπορεί να χρησιμοποιηθεί για την αυτόματη σύνθεση μιας μονάδας ZOLC σε επίπεδο RTL. Η περιγραφή αυτή έχει σχεδιαστεί ώστε να είναι εύκολα επαναστοχεύσιμη σε RISC επεξεργαστές με έκδοση μονής εντολής (single-issue) απαιτώντας πολύ μικρό ανθρώπινο κόπο για αυτό.

Η μονάδα ZOLC έχει ενσωματωθεί σε διάφορα μοντέλα RISC γενικού σκοπού αλλά και ΕΕΣ επεξεργαστών σε διαφορετικά επίπεδα αφαίρεσης (RTL VHDL και ArchC [ArchC]) χωρίς να επισημαίνεται μείωση στη μέγιστη συχνότητα χρονισμού του επεξεργαστή. Μια μέση βελτίωση των επιδόσεων (σε χρόνο εκτέλεσης) κατά 25.5% και 44% αντίστοιχα επιτεύχθηκαν για έναν ενσωματωμένο RISC γενικού σκοπού και έναν ΕΕΣ που στοχεύουν εφαρμογές-πυρήνες (εφαρμογίδια: kernel applications) όπως είναι οι εφαρμογές της εκτίμησης και αντιστάθμισης κίνησης (motion estimation and compensation) που απαντώνται στην συμπίεση βίντεο κατά τα πρότυπα MPEG (οι τελευταίες για τον ΕΕΣ). Μια αντίστοιχη βελτίωση της τάξης του 10% επιτεύχθηκε για τον προαναφερθέντα RISC γενικού σκοπού για ένα υποσύνολο των εφαρμογών δοκιμής MiBench [MiBench] στις οποίες δεν συναντώνται απαραίτητα οι παραπάνω, απαιτητικοί όσον αφορά τις επιδόσεις σε λειτουργίες βρόχων, μικροπυρήνες.

Το βασικό μειονέκτημα στην εξυπηρέτηση των λειτουργιών βρόχων τόσο σε μοντέρνους DSP (Motorola 56300, ST120, και TMS320C54x) όσο και σε ανανεωμένες αρχιτεκτονικές GPP (ARM, MIPS32) είναι ότι οι παρεχόμενες λύσεις, όταν υφίστανται, εστιάζουν στην εκτέλεση κανονικών (canonical) βρόχων οι οποίοι συναντώνται σε κλασικές DSP εφαρμογές. Έτσι, ένα μειονέκτημα των περισσότερων DSP της αγοράς είναι ότι είναι ικανοί στον χειρισμό μόνο τέλειων πλήρως φωλιασμένων δομών βρόχων οι οποίες αποτελούνται από στατικούς βρόχους μονής εισόδου. Προσδιορίσιμοι (configurable) επεξεργαστές όπως οι ARCTangent-A4 [ARC] και Xtensa-LX [Tensilica] είναι ακόμη περισσότερο περιοριστικοί καθώς περιλαμβάνουν μηχανισμούς μηδενικής επιβάρυνσης (βρόχων) για απλούς (συνήθων πρόκειται για βρόχους σε εσωτερη θέση) βρόχους μόνον. Στις περιπτώσεις αυτές, το αρχιτεκτονικό περίγραμμα δεν υποστηρίζει την προσθήκη εναλλακτικών για λειτουργίες ροής ελέγχου. Μια πλήρως λογισμική λύση στο ζήτημα της επιβάρυνσης των επιδόσεων λόγω των λειτουργιών ελέγχου βρόχου είναι το ξετύλιγμα βρόχων, το οποίο αποτελεί μια τεχνική βελτιστοποίησης μεταγλωττιστή που απομακρύνει την επιβάρυνση βρόχων με μικρό σχετικά αριθμό επαναλήψεων, που όμως δεν είναι εφαρμόσιμη σε περιπτώσεις που υφίστανται σε εσωτερικά επίπεδα φωλιάσματος σε μια δομή βρόχων λειτουργίες ελέγχου ροής προγράμματος ή σε περιπτώσεις με όρια βρόχων που να είναι άγνωστα (μη υπολογίσιμα) κατά την μεταγλώττιση της δομής βρόχου.

Η προτεινόμενη αρχιτεκτονική λύση ZOLC αντιμετωπίζει το πρόβλημα των εντολών για την εκτέλεση των λειτουργιών βρόχων στη γενική τους μορφή, επιτυγχάνοντας τα μέγιστα κέρδη όσον αφορά τις επιδόσεις. Η προτεινόμενη μονάδα ZOLC χρησιμοποιείται στο στάδιο διοχέτευσης ανάκλησης εντολής (instruction fetch stage) ενός επεξεργαστή και μπορεί να εφαρμοστεί για αυθαίρετα πολύπλοκες δομές βρόχων που μπορεί να αποτελούνται από

φυσικούς ή μη-μειώσιμους βρόχους με κόμβους πολλαπλών-εισόδων/πολλαπλών-εξόδων (multi-entry/multi-exit). Για την αξιοποίηση της μονάδας ZOLC αναπτύχθηκε μεταφερτή περιγραφή της, ένας φορμαλισμός των διεργασιών (tasks) που απαρτίζουν μια δομή βρόχων καθώς και μια πρακτική ροή μεταγλώττισης για την γέννηση ZOLC-συμβατού κώδικα, η οποία βασίζεται σε εργαλεία ανάπτυξης ανοικτού κώδικα. Επίσης, με τη τεχνική ZOLC, δεν απαιτείται η ύπαρξη πολύπλοκων περασμάτων μεταγλωττιστή για τη στόχευση του κώδικα (π.χ. ANSI C) των εφαρμογών, ενώ υποστηρίζονται και δομές ελέγχου με δυναμικούς βρόχους των οποίων οι τιμές των παραμέτρων (αρχική τιμή, τιμή βήματος, τελική τιμή) οριστικοποιούνται κατά την εκτέλεση.

Για την δοκιμή της τεχνικής ZOLC, μονάδες ZOLC ενσωματώθηκαν σε διάφορους RISC επεξεργαστές: στον XiRisc [Cam01], έναν 32-bit προσδιορισίμο μικροεπεξεργαστή που είναι διαθέσιμος σε VHDL πηγαίο κώδικα, όπως και σε γλώσσα περιγραφής αρχιτεκτονικής (ArchC) για άλλους επεξεργαστές (όπως ο MIPS R3000, και ένας ASIP για τις εφαρμογές εκτίμησης και αντιστάθμισης κίνησης).

3.1. Αναπαράσταση για εφαρμογές με εντατική επεξεργασία βρόχων

Η πληροφορία ροής ελέγχου για κάθε συνάρτηση σε μια δοθείσα εφαρμογή συλλαμβάνεται από το γράφο ροής ελέγχου (CFG), ο οποίος είναι ένας κατευθυντικός γράφος με καθέτους (κόμβους) που αναπαριστούν τα βασικά μπλοκ της συνάρτησης και ακμές που σημειώνουν την κατευθυντικότητα της ροής ελέγχου. Για την παραγωγή κώδικα από ενδιάμεση αναπαράσταση μεταγλωττιστή (IR) μορφής γράφου, απαιτείται η επεξεργασία του γράφου ροής ελέγχου-δεδομένων (CDFG) των συναρτήσεων της εφαρμογής, ο οποίος αποτελείται από τον CFG με τους κόμβους να επεκτείνονται στις εντολές που αποτελούν τα αντίστοιχα βασικά μπλοκ. Προκειμένου η αρχιτεκτονική ZOLC να εκτελεί τις διεργασίες βρόχου στο υπόβαθρο, ο μεταγλωττιστής θα πρέπει να επιτελεί τα παρακάτω βήματα:

- a) Απομάκρυνση των εντολών επιβάρυνσης βρόχων από το αρχικό CDFG
- b) Παραγωγή εντολών για την αρχικοποίηση των στοιχείων αποθήκευσης του ZOLC
- c) Την εισαγωγή εντολών για τη δυναμική ανανέωση των τιμών βήματος και ορίων βρόχου, και τον χειρισμό δυναμικών αποφάσεων ροής ελέγχου που προκύπτουν σε εξώτερες δομές ελέγχου if-then-else.

Για τον καθορισμό των τιμών των παραμέτρων για την βέλτιστη ρύθμιση (configuration) του ZOLC (αριθμός βρόχων, μέγιστος αριθμός εισόδων/εξόδων ανά βρόχο) κάτω από περιορισμούς επιφάνειας υλικού, υπονοείται η ανάγκη μιας διαδικασίας DSE με τις αντίστοιχες επαναλήψεις ώστε τον υπολογισμό όλων των σημείων που αποτελούν υποψήφιες λύσεις. Για την αποτίμηση ενός σημείου στο πεδίο λύσεων, η υπολογιστική πολυπλοκότητα που συνδέεται με την εκτέλεση των βημάτων a-c μπορεί να ελαττωθεί σημαντικά εκτελώντας τα b και c σε μια περισσότερο βολική αναπαράσταση γράφου για την εξεταζόμενη εφαρμογή.

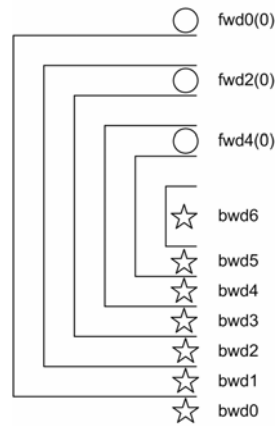
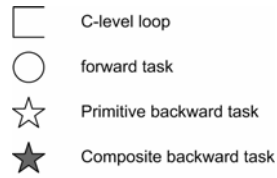
Η αναπαράσταση αυτή θα μπορούσε να μοντελοποιεί μόνο τις εκφράσεις μεταφοράς ελέγχου (Control Transfer Expressions – CTEs) ανάμεσα σε τμήματα κώδικα που είναι τοποθετημένα σε οριακές θέσεις σε ένα βρόχο. Οι (γενικά μη γραμμικές) λίστες των εντολών που απαρτίζουν αυτά τα τμήματα κώδικα είναι οι Διεργασίες Επεξεργασίας Δεδομένων (Data Processing Tasks – DPTs) του αλγορίθμου τον οποίο εκφράζει η εφαρμογή. Οι CTEs μοντελοποιούν αφαιρετικές λειτουργίες ελέγχου. Στην περίπτωση μας, οι CTEs είτε αντιστοιχούν σε σήματα του ZOLC σε επίπεδο υλικού είτε σε συνθήκες που εκδίδονται από εντολές του επεξεργαστή όπως συμβαίνει με τις δυναμικές λειτουργίες διακλάδωσης. Αυτή η δομή τύπου γράφου είναι ο Γράφος Ροής Ελέγχου Διεργασιών (Task Control Flow Graph – TCFG) και ορίζεται ακολούθως:

Ορισμός 3.1. Καλείται $TCFG(V \cup V', E)$ ο κατευθυντικός κυκλικός γράφος που αναπαριστά τη ροή ελέγχου σε μια αυθαίρετα σύνθετη δομή φωλιασμένων βρόχων ενός προγράμματος εφαρμογής· κάθε κόμβος V αναπαριστά ακριβώς μια στοιχειώδη (primitive) DPT, κάθε κόμβος V' μια σύνθετη DPT που προκύπτει ως αποτέλεσμα της εφαρμογής ενός τελεστή μετασχηματισμού που εξαρτάται από το υλικό σε ένα υποσύνολο των DPT· οι ακμές E αναπαριστούν εξαρτήσεις ελέγχου ανάμεσα στις διεργασίες επεξεργασίας δεδομένων.

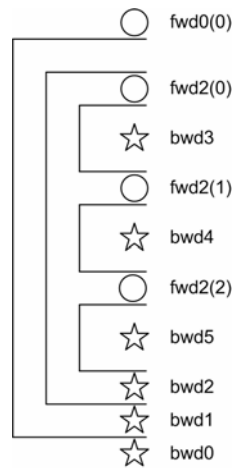
Οι διεργασίες οι οποίες δεν περιλαμβάνουν την ακολουθία (ή υπογράφο) εντολών επιβάρυνσης βρόχου προσδιορίζονται ως στοιχειώδεις εμπροσθοκατευθυντικές διεργασίες (P-FDPTs), ενώ αυτές που περιλαμβάνουν ακριβώς μια τέτοια ακολουθία ονομάζονται στοιχειώδεις οπισθοκατευθυντικές διεργασίες (P-BDPTs). Οι υπόλοιπες διεργασίες προσδιορίζονται ως σύνθετες διεργασίες (CDPTs) και εμφανίζονται ως αποτέλεσμα μετασχηματισμών γράφου οι οποίοι εφαρμόζουν κανόνες που εξαρτώνται από το υλικό. Μια τέτοια περίπτωση περιγράφεται διεξοδικά στην ενότητα 3.2.3.

Ως κινητήριο παράδειγμα, ας θεωρήσουμε τον πυρήνα εκτίμησης κίνησης πλήρους ανίχνευσης (full-search motion estimation – fsmc), ο οποίος χρησιμοποιείται στη συμπίεση MPEG για την απομάκρυνση μέρους του χρονικού πλεονασμού σε μια ακολουθία video. Ο αλγόριθμος αποτελείται από έξι φωλιασμένους βρόχους με το αντίστοιχο διάγραμμα εμφώλευσης βρόχων να απεικονίζεται στο Σχήμα 3.1.α και το TCFG στο Σχήμα 3.1.γ. Τα BDPTs σημειώνονται ως bwd_i όπου i είναι η τρέχουσα απαρίθμηση στη δομή βρόχων, ξεκινώντας από το 1 καθώς το μηδενικό επίπεδο δεσμεύεται να αποτελείται από τις προηγούμενες και διάδοχες δηλώσεις στη δομή. Τα CPDTs διακρίνονται από σήμανση συνεχούς περιοχής κατά $bwd_m - bwd_n$, όπως στο Σχ. 3.4 ή από σήμανση λίστας με $\{bwd_m, \dots, bwd_n\}$, με την τελευταία να χρησιμοποιείται για μη διαδοχικές BDPTs. Οι FDPTs σημειώνονται ως $fwd_i(j)$, όπου i είναι ο αύξων αριθμός βρόχου και το j επιλέγει μια συγκεκριμένη διεργασία του τύπου αυτού από τον i -th βρόχο. Αυτός ο φορμαλισμός για το διαχωρισμό των τύπων διεργασιών χρησιμοποιείται από τις εσωτερικές δομές δεδομένων ενός περάσματος μεταγλωττιστή για τον σχηματισμό TCFGs. Το σήμα `loopend` καταγράφει μια συνθήκη τερματισμού βρόχου κατά το χρονικό σημείο που η εκτέλεση βρίσκεται σε μια bwd διεργασία. Το σήμα θα πρέπει να παράγεται από τη μονάδα ZOLC για την οδήγηση της μεταγωγής διεργασιών.

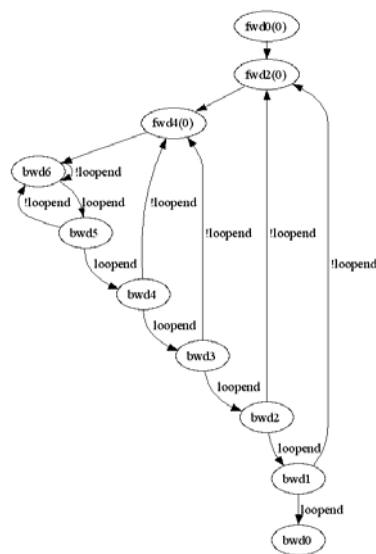
Τα BDPTs συμβολίζονται ως αστέρια στα Σχήματα 3.1.α-β και μπορούν να τοποθετηθούν σε εσωτερικές ή τερματικές θέσεις σε ένα βρόχο, ενώ η τελική διεργασία του τύπου αυτού, bwd_0 , σημειώνει το σημείο εξόδου από τη δομή βρόχων. Οι δείκτες βρόχων ανανεώνονται κατά τη διάρκεια της εκτέλεσης αυτών των διεργασιών. Τα FDPTs συμβολίζονται με κύκλο και είναι τοποθετημένα σε μη-τερματικές θέσεις σε ένα βρόχο. Τέτοιες διεργασίες μπορούν να συμμετέχουν σε αποφάσεις αλλαγής ροής ελέγχου και δεν επηρεάζουν τους δείκτες βρόχων.



(α)



(β)



(γ)

Σχήμα 3.1: Δομές βρόχων και όψεις TCFG αλγοριθμικών πυρήνων με εντατική επεξεργασία βρόχων. Για τις CTEs, χρησιμοποιείται ANSI C σημειογραφία. (α) Διάγραμμα εμφώλευσης βρόχων για μια πλήρως φωλιασμένη δομή (fsme). (β) Μια σύνθετη δομή βρόχων που συναντάται σε διορθογώνιο φίλτρο κυματιδίου (biorthogonal wavelet filter). (γ) TCFG για τον αλγόριθμο fsme.

3.2. Περιπτώσεις χρήσης μιας μηχανής ZOLC

3.2.1. Περίπτωση μελέτης: Πολλαπλασιασμός πινάκων κατά μπλοκ σε ένα DSP με λειτουργίες ZOLC

Ως ένα τυπικό παράδειγμα χρήσης ελέγχου ZOLC θέτουμε έναν υποθετικό (για τον οποίο όμως υπάρχει λειτουργικό μοντέλο ακρίβειας κύκλου) DSP επεξεργαστή, ονόματι hDSP. Ο hDSP διαθέτει ετερογενή οργάνωση αποθήκευσης (δυαδική συστοιχία μνήμης, εξειδικευμένους καταχωρητές), πολλαπλασιασμό-και-συσσώρευση ενός κύκλου, γεννήτρια διευθύνσεων (AGU) και αρχιτεκτονική βαθμίδων διοχέτευσης με 5 στάδια εμπνευσμένη από τον κλασικό σχεδιασμό του DLX.

Μια πληθώρα από εφαρμογές πολυδιάστατης επεξεργασίας σήματος (multi-dimensional signal processing) όπως η μετατροπή χρωματικού πεδίου, οι μετασχηματισμοί πεδίου συχνότητας-χρόνου, και το φιλτράρισμα εικόνας, συνεπάγονται την επεξεργασία σε στοιχειώδεις πολλαπλασιασμούς πίνακα-με-πίνακα, οι οποίοι υλοποιούνται από τριπλά φωλιασμένους βρόχους που περιέχονται σε άλλους εξώτερους βρόχους για το σκανάρισμα των μπλοκ που πολλαπλασιάζονται. Ένας τέτοιος πυρήνας είναι κρίσιμος για τις επιδόσεις της εφαρμογής που τον περιέχει και για αυτό είναι συνήθως συντεταγμένος σε γλώσσα συμβολομεταφραστή του επεξεργαστή. Στο Σχ. 3.2.α δίνεται ο κώδικας αναφοράς σε C. Το Σχ. 3.2.β δείχνει το βελτιστοποιημένο τμήμα κώδικα για την εφαρμογή-πυρήνα όπου με AGU συμβολίζονται οι ακέραιες πράξεις στη μονάδα παραγωγής διευθύνσεων, '@' είναι η τρέχουσα διεύθυνση για την προσπέλαση μιας μεταβλητής τύπου πίνακα σε συστοιχία μνήμης (X ή Y), '|' χρησιμοποιείται για τις παράλληλα εκτελούμενες μικρολειτουργίες, RI, RF, RX, RP είναι οι κατηγορίες καταχωρητών για την αρχική και τελική παράμετρο βρόχου, την τρέχουσα τιμή δείκτη και για τις γενικές λειτουργίες, και ACC είναι ο καταχωρητής συσσώρευσης.

Για λειτουργίες διακλάδωσης με απαίτηση 3 κύκλων, η συνολική απαίτηση σε αριθμό κύκλων για τον εσώτερο βρόχο του Σχήματος 3.2.β είναι 10 κύκλοι μηχανής. Το πλεονέκτημα της χρήσης του ZOLC είναι ότι οι λειτουργίες της αύξησης δείκτη και σύγκρισης-και-διακλάδωσης (που χρειάζονται 4 κύκλους) μπορούν να απομακρυνθούν και να αντικατασταθούν από λειτουργία μεταγωγής διεργασίας συνδεδεμένη με την ανάκληση της τελευταίας χρήσιμης εντολής στον εσώτερο βρόχο (MAC ACC, @CEOFF, @IMAGE). Όταν χρησιμοποιείται η μονάδα ZOLC, ο εσώτερος βρόχος εκτελείται σε 6 κύκλους, το οποίο σημαίνει κατά προσέγγιση 40% βελτίωση στις επιδόσεις κύκλων μηχανής. Γενικά, η πληροφορία που δεν είναι διαθέσιμη κατά το χρόνο μεταγλώττισης πρέπει να καταχωρείται στα στοιχεία αποθήκευσης του ZOLC κατά την εκτέλεση (at runtime), κάτι που αποκαλύπτει την ανάγκη για εντολές επέκτασης για τους επεξεργαστές που σχεδιάζονται για αυτό το σκοπό. Τέτοιοι βρόχοι ονομάζονται ημιστατικοί και δεν μπορούν να ξετυλιχτούν κατά το χρόνο μεταγλώττισης. Είναι κοινοί σε ορισμένες εφαρμογές όπως το διορθογώνιο (2,2) Cohen-Daubencies-Feauveau φίλτρο κυματιδίου (Σχ. 3.1.β) που χρησιμοποιείται στην συμπίεση εικόνας με Βαθμωτό Κβαντισμό Κυματιδίου (Wavelet-Scalar Quantization – WSQ) [Kum00].

Για λειτουργία που ελέγχεται από ZOLC, η πληροφορία που σχετίζεται με τις διεργασίες πρέπει να είναι αποθηκευμένη σε τοπικά αρχεία καταχωρητών. Τα δεδομένα που αποδίδονται σε κάθε διεργασία αποτελούνται από τα ακόλουθα στοιχεία, τα οποία δίνονται στην μορφή της εγγραφής (record) TaskContext του Σχήματος 3.2.γ.:

- Η διεύθυνση εισόδου απαριθμητή προγράμματος (PC entry address) η οποία μπορεί να είναι σχετική ή απόλυτη ανάλογα με τη συγκεκριμένη ZOLC υλοποίηση. Για διεργασίες πολλαπλών εισόδων υφίστανται περισσότερες της μιας τέτοιες διευθύνσεις.
- Η διεύθυνση εξόδου PC, δηλαδή η διεύθυνση της τελευταίας χρήσιμης υπολογιστικής εντολής στη διεργασία. Παρόμοια με παραπάνω, οι διεργασίες πολλαπλών-εξόδων μπορούν να έχουν περισσότερες από μία πιθανές διευθύνσεις εξόδου PC.
- Οι κωδικοποιήσεις για τα πιθανά διάδοχα DPTs. Η απόφαση λαμβάνεται κατά το χρόνο εκτέλεσης με παρατήρηση της συνθήκης τερματισμού βρόχου (για bwd διεργασίες) ή πιθανόν μιας συνθήκης δυναμικής εμπροσθοκατευθυντικής διακλάδωσης (για fwd διεργασίες).
- Η διεύθυνση βρόχου στην οποία αντιστοιχούνται πιθανές διάδοχες διεργασίες.
- Ο τύπος διεργασίας (fwd/bwd) για αυτές τις διεργασίες.

Το Σχήμα 3.2.δ απεικονίζει τις τιμές περιεχομένου (context values) για τις πιο σημαντικές διεργασίες του αλγορίθμου matmult: fwd4(0), bwd5 και bwd4.

```
INPUT
image : [0..N*M-1] array
coeff : [0..B-1,0..B-1] array
OUTPUT
output: [0..N*M-1] array

matmult()
{
  ...
  For (i=0; i<N; i+=B) {
    For (j=0; j<M; j+=B) {
      For (k=0; k<B; k++) {
        For (l=0; l<B; l++) {
          acc = 0;

          For (m=0; m<B; m++) {
            acc += coeff[k][m]*image[(i+m)*M+(j+l)];}

          output[(i+k)*M+(j+l)] = acc;
        } } } }
}
```

(α)

```

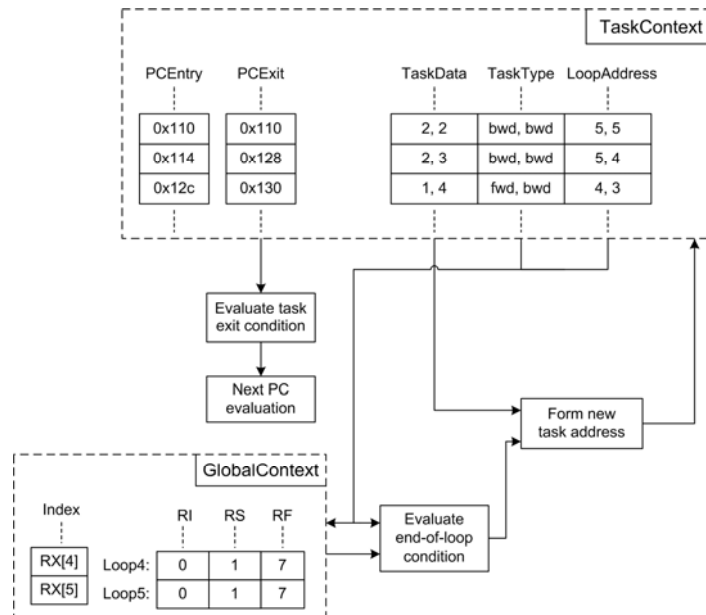
loop4: # fwd4(0)
      # acc = 0
      ACC = 0
      RI[5] = 0
loop5: # bwd5
      # acc += coeff[k+B*m] * image[(i+m)+M*(j+1)]
      AGU A.Y.2 = RX[2] + RX[4]
      AGU A.Y.2 = A.Y.2 * RP[2]
      AGU A.X.2 = RX[5] * RP[5]
      AGU A.X.1 = RX[3] || A.Y.1 = RX[1] + RX[5]
      LOD @COEFF, A.X || LOD @IMAGE, A.Y
      MAC ACC, @COEFF, @IMAGE
      INC RX[5], #1
      BNE RX[5], RF[5], loop5
      # bwd4
      # output[(i+k)+M*(j+1)] = (uchar)(acc>>8)
      AGU A.Y.1 = RX[1] + RX[3]
      LOD @OUTPUT, A.Y
      STR ACC, @OUTPUT
      INC RX[4], #1
      BNE RX[4], RF[4], loop4
    
```

(β)

```

record TaskContext is
  PCEnterAddress : data_vector(0:PCW-1)
  PCExitAddress  : data_vector(0:PCW-1)
  record TaskSwitchingEntry is
    TaskData      : data_vector(0:log2(Nt)-1, 0:1)
    TaskTypeSelect: data_bit(0,0:1)
    LoopAddress   : data_vector(0:log2(Nl)-1, 0:1)
  end record
end record
    
```

(γ)



(δ)

Σχήμα 3.2: Αποτύπωση εφαρμογής κατά τη τεχνική ZOLC για τον αλγόριθμο πολλαπλασιασμού πινάκων κατά μπλοκ (matmult). (α) ANSI C κώδικας αναφοράς για τον matmult. (β) hDSP κώδικας συμβολομεταφραστή για τον matmult που έχει βελτιστοποιηθεί χειρωνακτικά από τον ερευνητή. (γ) Η εγγραφή TaskContext περιέχει την εγγενή πληροφορία

που επισυνάπτεται σε μια διεργασία επεξεργασίας δεδομένων (DPT). (δ) Οι τιμές των TaskContext και GlobalContext για διεργασίες του αλγορίθμου που είναι κρίσιμες για την συνολική επίδοση του matmult.

Η αρχικοποίηση του ZOLC για την προετοιμασία του επεξεργαστή για την είσοδο σε κατάσταση όπου όλες οι λειτουργίες βρόχων ελέγχονται από το ZOLC μπορεί να πραγματοποιηθεί από ακολουθία εντολών που παράγεται από πέρασμα μεταγλωττιστή το οποίο επεξεργάζεται την TCFG αναπαράσταση (ενότητα 3.4). Οι απαιτούμενες εντολές επέκτασης για αυτό είναι αντικείμενο συζήτησης στην ενότητα 3.4.2. Για ολόκληρο τον αλγόριθμο matmult (ο οποίος διαθέτει 5 φωλιασμένους βρόχους) οι απαιτούμενοι κύκλοι αρχικοποίησης υπολογίζονται ως:

- 16 εντολές για την αποθήκευση των απολύτων (ή σχετικών) διευθύνσεων PC εισόδου και εξόδου για την επισήμανση των ορίων των διεργασιών
- 14 εντολές για την αρχικοποίηση της πληροφορίας μεταγωγής διεργασιών όταν η εκτέλεση έχει προχωρήσει στην αντίστοιχη δομή βρόχων. Στην περίπτωση αυτή, το περιεχόμενο μιας διεργασίας αναπαρίσταται από την συνένωση της απαρίθμησης της διεργασίας (task enumeration) και της τιμής της συνθήκης τερματισμού βρόχου για τον τρέχοντα βρόχο (ο οποίος διαθέτει πιθανώς πολλαπλά σημεία εξόδου). Η πληροφορία μεταγωγής διεργασιών αποτελείται από την απαρίθμηση διεργασίας, τον τύπο διεργασίας (task type) και τη διεύθυνση βρόχου (loop address) για μια πιθανή διάδοχη διεργασία.
- 15 εντολές για τη διαμόρφωση της αρχικής, τελικής και τιμής βήματος ενός βρόχου.
- Μια επιπρόσθετη εντολή χρειάζεται για τον προσδιορισμό ενός καταχωρητή-μάσκα που χρησιμοποιείται για την ρύθμιση των εκφράσεων δείκτη που είναι συνδεδεμένες με κάθε bwd διεργασία στην αριθμητική λογική μονάδα (ALU) υπολογισμού δείκτη. Εφόσον δεν αναφέρεται διαφορετικά, οι λειτουργίες της ALU αυτής είναι πρόσθεση και αφαίρεση.

Ο συνολικός αριθμός των απαιτούμενων κύκλων (46) είναι αμελητέος σε σύγκριση με τους άνω των 5.7 εκατομμυρίων που χρειάζονται για τον matmult σε CIF (352x288) εικόνες (για την συνιστώσα φωτεινής έντασης Υ) όταν γίνεται η διαδικασία κατά πίνακες 8x8. Για έναν αλγόριθμο με 20 βρόχους (τέτοιος είναι ο fsme_dr του Πίνακα 3.4), οι κύκλοι επιβάρυνσης λόγω της αρχικοποίησης περιορίζονται σε 164, το οποίο είναι κατά τρεις τάξεις μεγέθους (τουλάχιστον) λιγότερο από οποιαδήποτε άλλη μέθοδο συμπεριλαμβανομένης και της [Tal03]. Επίσης, η προσέγγιση ZOLC έχει και άλλες θετικές συνέπειες: βελτιώνει την απόκριση του επεξεργαστή σε περιορισμούς πραγματικού χρόνου (real-time constraints) καθώς μειώνει τη διαφορά μεταξύ της βέλτιστης και χειρίστης περίπτωσης χρόνου εκτέλεσης (BCET-WCET). Επιπρόσθετα, η χρήση του ελέγχου ZOLC είναι ακόμη πιο σημαντική για αρχιτεκτονικές με παράλληλη επεξεργασία δεδομένων και εσώτερους βρόχους μικρής έκτασης. Για την αρχιτεκτονική MorphoSys που περιλαμβάνει ένα διδιάστατο πίνακα από επαναπροσδιορίσιμα στοιχεία, η ομάδα ανάπτυξης του αναφέρει βαθμό επιτάχυνσης στην εκτέλεση εφαρμογών της τάξης του 4 αποκλειστικά εξαιτίας της μηδενικής επιβάρυνσης βρόχων για συχνά εκτελούμενες διεργασίες, όπως αυτές που περιέχουν λειτουργίες μεταφοράς δεδομένων κατά μπλοκ [Pan03].

3.2.2. Υποστήριξη βρόχων πολλαπλών εισόδων για λειτουργία ZOLC

Οι βρόχοι πολλαπλών εισόδων αποτελούν μη-μειώσιμες περιοχές του CFG, οι οποίες δεν είναι ανιχνεύσιμες από τον κλασσικό αλγόριθμο ανάλυσης φυσικών βρόχων [Aho86] που

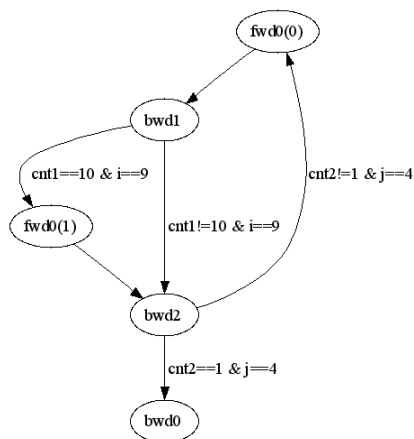
χρησιμοποιείται από αρκετούς επαναστοχεύσιμους μεταγλωττιστές [Ben88],[Smi02],[Sta02]. Η βασική συνέπεια αυτού είναι ότι για τις περιοχές αυτές δεν μπορούν να εφαρμοστούν οικείες τεχνικές βελτιστοποίησης κώδικα. Η κύρια προσέγγιση για να ξεπεραστεί αυτό είναι η χρησιμοποίηση της τεχνικής διαχωρισμού κόμβων (node splitting) [Aho86] που θεωρητικά αυξάνει σε εκθετικό βαθμό το μέγεθος κώδικα και για αυτό αποφεύγεται από την πλειοψηφία των μεταγλωττιστών που χρησιμοποιούνται για ερευνητικούς σκοπούς. Βελτιωμένες εκδοχές του διαχωρισμού κόμβων έχουν προταθεί πρόσφατα [Ung02] οι οποίες χρησιμοποιούν έναν ειδικό τύπο γράφου για την ανάθεση μοναδικών κλάσεων στις οπισθόδρομες ακμές ενός CFG. Ένα μειονέκτημα είναι ότι η τεχνική αυτή επιβάλλει την ανάγκη για εκτεταμένες τροποποιήσεις στους υπάρχοντες μεταγλωττιστές ώστε να λαμβάνουν υπόψη τέτοιου είδους αναπαράσταση.

Προκειμένου τη διερεύνηση βρόχων πολλαπλών εισόδων για ενδεχόμενη λειτουργία κατά ZOLC, η αναπαράσταση TCFG του Σχήματος 3.3.β επαρκεί για να μοντελοποιήσει επακριβώς τη δομή βρόχων του κώδικα του Σχήματος 3.3.α. Έτσι, οι πρέπουσες εντολές θα απομακρυνθούν από όλες τις bwd διεργασίες και τις fwd που περιέχουν αλληλουχίες εντολών για την αρχικοποίηση παραμέτρων βρόχου. Έπειτα από αυτό, με απλή απαρίθμηση ακμών, μπορεί να εξαχθεί η απαραίτητη πληροφορία μεταγωγής/διαδοχής των διεργασιών για την αρχικοποίηση του ZOLC.

Η TCFG είναι μια ομογενής αναπαράσταση για σύνθετες δομές βρόχων, και συγκρινόμενη με τις καλύτερες μεθόδους για τον χειρισμό των μη-μειώσιμων βρόχων, όχι μόνο επιτυγχάνεται η αποφυγή της εκτεταμένης αύξησης του κώδικα, αλλά μειώνονται τόσο το μέγεθος κώδικα όσο και ο αριθμός των δυναμικών εντολών που εκτελούνται για την εφαρμογή.

```
L1: cnt1++;  
  
for (i=1; i<10; i++) {  
    a[i] = a[i-1] + 1;  
  
    if (cnt1==10 & i==9)  
        break;  
    else if (cnt1!=10 & i==9)  
        goto L2; }  
  
cnt2++;  
  
L2: for (j=0; j<5; j++) {  
    b[j] = 2*a[j]-1;  
  
    if (cnt2==1 & j==4)  
        break;  
    else if (cnt2!=1 & j==4)  
        goto L1; }
```

(α)

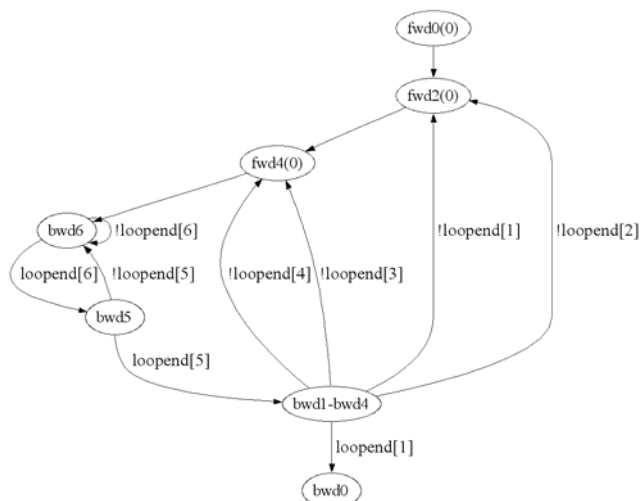


(β)

Σχήμα 3.3: Αντιπροσωπευτικός πηγαίος κώδικας σε C και το αντίστοιχο του CFG για μια περίπτωση βρόχων πολλαπλών εισόδων. (α) Πηγαίος κώδικας σε C για το παράδειγμα. (β) Το αντίστοιχο CFG.

3.2.3. Υποστήριξη για εναλλακτικά μοντέλα λειτουργιών βρόχων στο υλικό (hardware looping)

Μια ενδιαφέρουσα εφαρμογή της παρουσιαζόμενης τεχνικής ελέγχου ZOLC είναι η συνεργιστική της χρήση με άλλες τεχνικές υλοποίηση λειτουργιών βρόχων στο υλικό. Μια τέτοια περίπτωση είναι η υποστήριξη της ανανέωσης πολλαπλών δεικτών μέσα σε ένα κύκλο ρολογιού [Tal03],[HWLU]. Για την εφαρμογή fsmc, τέσσερις διαδοχικές bwd διεργασίες μπορούν να συνενωθούν στην σύνθετη διεργασία 'bwd1-bwd4' όπως αυτό φαίνεται στο Σχ. 3.4. Έτσι, απαιτείται συνολικά ένας αριθμός από σήματα τερματισμού βρόχου ίσος με τον αριθμό των βρόχων στον fsmc. Στο Σχ. 3.5 απεικονίζονται σε μορφή ψευδοκώδικα οι κανόνες που υλοποιούν τις προαναφερθείσες (στοιχειώδη και σύνθετη) διεργασίες. Στην τελευταία περίπτωση, η αντίστοιχη μονάδα μετασχηματισμού είναι ικανή να αναδομεί ένα δοσμένο CFG με τη συνένωση διαδοχικών PB-DPTs.



Σχήμα 3.4: CFG για μια υλοποίηση του fsmc που χρησιμοποιεί την υλική μονάδα ανανέωσης δείκτη του [Tal03].

```
PrimitiveBackwardTask()
begin
  useful computation operations in this task;
  if index[m] equals loop-final[m] then
    reset index[m] to loop-initial[m]
  else
    increment index[m]
  endif
end

CompositeBackwardTask()
begin
  useful computation operations in this task;
  if index[m] equals loop-final[m] then
    index[m].index[n] := loop-initial[m].loop-initial[n]
  elsif index[m+1] equals loop-final[m+1] then
    increment index[m]
    index[m+1].index[n] := loop-initial[m+1].loop-initial[n]
    ...
  elsif index[n] equals loop-final[n] then
    increment index[n-1]
    index[n] := loop-initial[n]
  else
    increment index[n]
  endif
end
```

Σχήμα 3.5: Ανανέωση δείκτη σε στοιχειώδεις (PrimitiveBackwardTask) και σύνθετες (CompositeBackwardTask) οπισθοκατευθυντικές διεργασίες, αντίστοιχα.

3.3. Ενσωμάτωση της μονάδας ZOLC σε προγραμματιζόμενους επεξεργαστές

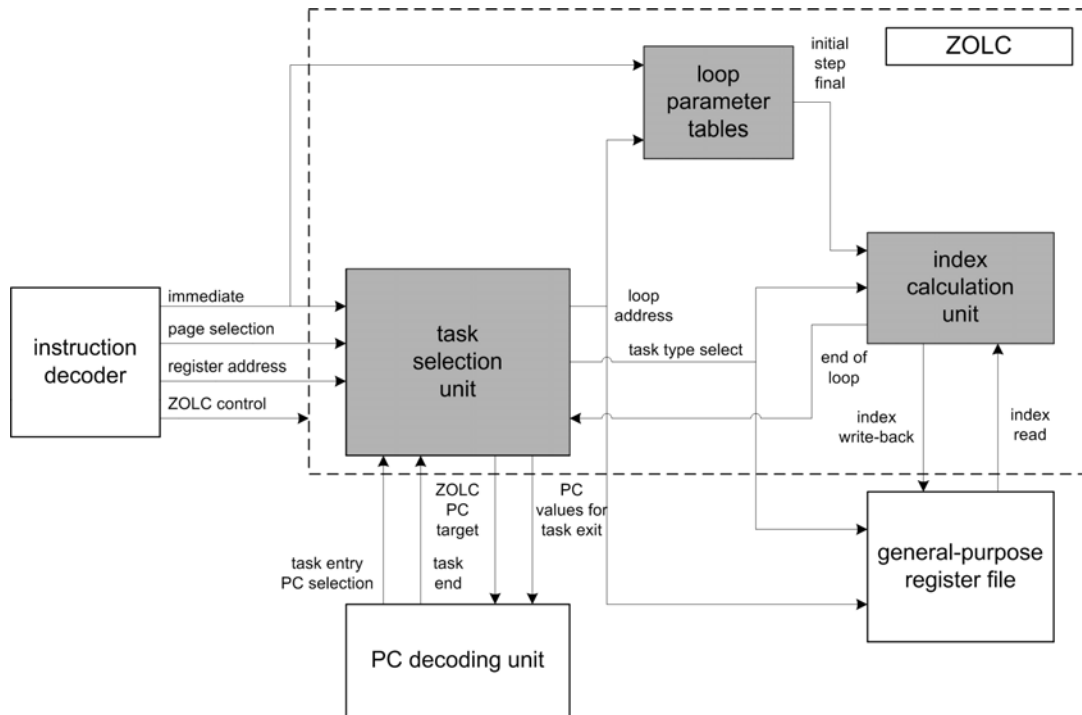
3.3.1. Άποψη περιγράμματος ενός επεξεργαστή ενισχυμένου από λειτουργίες ZOLC

Ένα διάγραμμα βαθμίδων που απεικονίζει το πώς η προτεινόμενη αρχιτεκτονική ZOLC μπορεί να ενσωματωθεί στο διάδρομο ελέγχου ενός τυπικού RISC επεξεργαστή φαίνεται στο Σχ. 3.6. Ο σκοπός του ZOLC είναι να παρέχει την πρότερη υποψήφια διεύθυνση PC στην μονάδα αποκωδικοποίησης του PC για κάθε λειτουργία βρόχου που έχει αντικατασταθεί ως προς τον αρχικό κώδικα της εφαρμογής. Τυπικά, ο σχεδιασμός του αποκωδικοποιητή εντολών, της μονάδας αποκωδικοποίησης PC, και της αρχιτεκτονικής αρχείου καταχωρητών απαιτεί τροποποιήσεις λόγω της παρουσίας του ZOLC υλικού. Η μονάδα ZOLC αποτελείται από την μονάδα επιλογής διεργασίας (task selection unit), που καθορίζει το ποια είναι η κατάλληλη επόμενη τιμή του PC όταν η εκτέλεση βρίσκεται σε μια δομή βρόχων, οι πίνακες αναζήτησης παραμέτρων βρόχων (loop parameter tables) όπου διατηρούνται οι παράμετροι βρόχου και η μονάδα υπολογισμού δείκτη.

Από την μεριά του ZOLC διακρίνονται δύο τρόποι λειτουργίας του. Στην λειτουργία «αρχικοποίησης», οι πόροι αποθήκευσης του ZOLC αρχικοποιούνται από εντολές που χρησιμοποιούν μια εξειδικευμένη διαμόρφωση για προσπέλαση σελίδας (paged-access) οι οποίες πρέπει να έχουν προστεθεί στο ρεπερτόριο εντολών του επεξεργαστή. Στην «ενεργό» λειτουργία, η μονάδα ZOLC:

- Καθορίζει την ακόλουθη διεργασία
- Εκδίδει μια νέα τιμή-στόχο για τον PC και ένα σύνολο από υποψήφιες τιμές διεύθυνσης εξόδου (για την περίπτωση βρόχων πολλαπλών-εξόδων) προς τη μονάδα αποκωδικοποίησης PC
- Ανανεώνονται οι δείκτες βρόχων και γράφονται είτε σε διακεκριμένους καταχωρητές του γενικού αρχείου καταχωρητών είτε σε μια ξεχωριστή συστοιχία καταχωρητών δείκτη.

Η πληροφορία της αλληλουχίας διεργασιών αποθηκεύεται σε έναν πίνακα αναζήτησης (LUT) εντός της μονάδας επιλογής διεργασιών. Με την ολοκλήρωση μιας DPT, εκδίδεται σήμα τερματισμού διεργασίας (task end) από τη μονάδα αποκωδικοποίησης PC και επιλέγεται μια καταχώρηση από το LUT για τη διευθυνσιοδότηση της διάδοχης DPT και των παραμέτρων βρόχου, με βάση το ποια διεργασία μόλις ολοκληρώθηκε και την κατάσταση του τρέχοντος βρόχου. Οι παράμετροι βρόχου χρησιμοποιούνται για τον υπολογισμό της τρέχουσας τιμής δείκτη και τον καθορισμό του αν ο βρόχος έχει τερματιστεί.



Σχήμα 3.6: Ενσωματώνοντας την αρχιτεκτονική ZOLC σε προγραμματιζόμενους RISC επεξεργαστές.

3.3.2. Αρχιτεκτονικές λεπτομέρειες για τα στοιχεία του ZOLC

Όπως φαίνεται από το Σχ. 3.6, η μονάδα ZOLC δέχεται ένα διάνυσμα ελέγχου από τον τροποποιημένο αποκωδικοποιητή εντολών, το οποίο αποτελείται από σήματα επίπτωσης εγγραφής για τα αρχεία καταχωρητών της, την επιλογή σελίδας και τη διεύθυνση για την επιλογή ενός συγκεκριμένου καταχωρητή. Αυτή είναι η απαιτούμενη πληροφορία ελέγχου την οποία η μονάδα επιλογής διεργασίας, οι πίνακες παραμέτρων βρόχου, και το τροποποιημένο αρχείο καταχωρητών δέχονται από τις αποκωδικοποιημένες εντολές. Για την περίπτωση του RISC επεξεργαστή XiRisc, το διάνυσμα αποκωδικοποίησης εντολής επεκτείνεται για την παραγωγή $kent+kext+7$ επιπρόσθετων bit για τον χειρισμό των πόρων του ZOLC, όπου kent και kext είναι ο μέγιστος επιτρεπόμενος αριθμός εισόδων/εξόδων (αντίστοιχα) για κάθε βρόχο.

Αναφορικά με τους καταχωρητές δείκτη σε έναν επεξεργαστή που χρησιμοποιεί μονάδα ZOLC, υφίστανται δύο πιθανότητες: οι δείκτες να εκχωρούνται ως μια ξεχωριστή κλάση καταχωρητών δεσμεύοντας υποσύνολο του αρχείου καταχωρητών γενικού σκοπού ή μια (αποκλειστική) συστοιχία καταχωρητών δείκτη που θα πρέπει να εισαχθεί ως στοιχείο της αρχιτεκτονικού περιγράμματος του επεξεργαστή. Η πρότερη περίπτωση προτιμάται καθώς δεν απαιτεί εντολές επέκτασης για την αρχικοποίηση του ZOLC. Στην πράξη, χρησιμοποιήθηκαν για τον hDSP, μια κλάση 16 καταχωρητών στο γενικό αρχείο καταχωρητών 64 θέσεων του, και αντίστοιχα 8 δείκτες στον XiRisc (32 θέσεις στη μονάδα

επιλογής διεργασίας) χωρίς αρνητική επίπτωση στις επιδόσεις των εφαρμογών δοκιμής που εξετάστηκαν, συμπέρασμα που ενισχύεται από τα αποτελέσματα ανάλυσης χρόνου ζωής καταχωρητών που ελήφθησαν με τον Machine-SUIF. Αν ακολουθηθεί η προσέγγιση της συστοιχίας καταχωρητών δείκτη, απαιτούνται 3 επιπρόσθετα bit στις κωδικοποιήσεις των εντολών, κάτι που μπορεί και να μην είναι εφικτό. Οι πίνακες παραμέτρων βρόχου χρησιμοποιούνται για την αποθήκευση των οριακών τιμών και της τιμής βήματος (stride) των βρόχων. Τιμές αυτών που είναι άγνωστες κατά το χρόνο μεταγλώττισης είναι διαχειρίσιμες με δυναμικές ανανεώσεις. Έτσι, σχεδιάζεται ένας δεύτερος τμηματικός διάδρομος δεδομένων για τη μεταφορά του περιεχομένου των καταχωρητών γενικού σκοπού στους αντίστοιχους σελιδοποιημένους καταχωρητές. Απλές εντολές μετακίνησης (move) χρησιμοποιούνται για την ανανέωση των παραμέτρων βρόχου κατά τη φυσιολογική ροή προγράμματος.

Η μονάδα επιλογής διεργασίας ενσωματώνει ένα τετμημένο LUT για την επιλογή της διάδοξης διεργασίας, διεύθυνσης βρόχου και τύπου διεργασίας. Ο αριθμός των καταχωρήσεων σε αυτό το LUT είναι $2(nl-1)$ όπου nl είναι ο μέγιστος αριθμός διεργασιών που υποστηρίζει σε μια δομή βρόχων. Στην πιο ακραία περίπτωση, $nl=2nl+1$ όπου nl είναι ο μέγιστος αριθμός βρόχων που υποστηρίζεται σε μια συγκεκριμένη υλοποίηση του ZOLC. Το μήκος της λέξης δεδομένων για αυτά τα LUT είναι $\log_2(nl) + 1$, $\log_2(nl)$ και 1, αντίστοιχα, καθώς είναι ανούσιος ο προσδιορισμός μεταγωγής διεργασιών με το $bwd0$ ως πηγή. Έτσι, ο αριθμός των διεργασιών που είναι ικανά να οδηγήσουν τη μεταγωγή διεργασιών δίνεται από τη σχέση $nl' = 2 \times nl$. Για απρόσημα άμεσα ορίσματα των 16-bit, μια καταχώρηση για κάθε DPT ενός αλγορίθμου των 128 βρόχων θα μπορούσε να ανατεθεί με μία και μόνο εντολή.

Επίσης, (προσέγγιση που ακολουθήθηκε σε αντίστοιχα μοντέλα προσομοίωσης σε SystemC αλλά και στον VHDL κώδικα) `keyt` εξειδικευμένα αρχεία καταχωρητών για τον καθορισμό του PC στόχου για τη διάδοξη διεργασία και οι `keyxt` υποψήφιας τιμές PC για την συνθήκη πολλαπλής εξόδου θα πρέπει να είναι τοποθετημένα στην ίδια μονάδα. Αν δεν είναι απαραίτητη η υποστήριξη βρόχων πολλαπλών εξόδων, τότε `keyt = keyxt = 1`. Η επιβάρυνση για την αποθήκευση πολλαπλών τιμών για τον PC που συνήθως δεν χρησιμοποιούνται μπορεί να αντιμετωπιστεί με μια βελτιωμένη υλοποίηση που αξιοποιεί καταχωρήσεις-κλειδιά για τις διευθύνσεις PC εισόδου/εξόδου μιας διεργασίας. Σε κάθε διεργασία ανατίθεται ένα διπλό κλειδί (`keyent`, `keyext`) επισημαίνοντας τον αριθμό εισόδων και εξόδων για τη συγκεκριμένη διεργασία. Οι πραγματικές διευθύνσεις PC τότε προσπελαύνονται σε αυξητικές διευθύνσεις από το κλειδί. Ο PC στόχος καθορίζεται μέσα από `keyxt` πιθανές επιλογές με βάση την τιμή του διανύσματος επιλογής PC εισόδου διεργασίας όπως δείχνεται στο Σχήμα 3.6. Οι απλοί συγκριτές ισότητας που χρησιμοποιούνται για τον υπολογισμό του αποτελέσματος της συνθήκης πολλαπλής εξόδου ανήκουν στη μονάδα αποκωδικοποίησης PC.

Η μονάδα υπολογισμού δείκτη αποτελείται από ένα FSM δύο καταστάσεων για την γνωστοποίηση της κατάστασης του τρέχοντος βρόχου (πρώτη ή ακόλουθη επανάληψη) και από τη συνδυαστική υπομονάδα ανανέωσης δείκτη. Η ανανέωση δείκτη πραγματοποιείται μόνο με την ολοκλήρωση μιας `bwd` διεργασίας. Επίσης, το γενικού σκοπού αρχείο καταχωρητών τροποποιείται ώστε να υποστηρίζει μια θύρα ανάγνωσης και μια εγγραφής αφιερωμένες στις μεταφορές δεικτών βρόχων.

Ο αριθμός των bit αποθήκευσης για τον προσδιορισμό του ZOLC μπορεί να εξαχθεί εύκολα από την σύνοψη των μονάδων που εισάγονται στην αρχιτεκτονική του επεξεργαστή λόγω του ZOLC και οι οποίες δίνονται αναλυτικά στον Πίνακα 3.1. Στην ακόλουθη έκφραση, η απαρίθμηση των bit αποθήκευσης υπολογίζεται αθροίζοντας την συνεισφορά της κάθε μονάδας όπως αυτή δίνεται από το γινόμενο της ποσότητας αντιτύπων (Number of instances: Quantity) με τον αριθμό καταχωρήσεων (Num. Entries) και τα bit ανά καταχώρηση (bits per entry):

$$\sum_{Units} (\text{Quantity}) \times (\text{Num. entries}) \times (\text{Bits per entry}) =$$
$$2 \cdot (4 \cdot n_l + 1) \cdot (\log_2(n_l) + 1) + 2 \cdot (k_{ent} + k_{ext}) \cdot n_l \cdot PCW$$
$$+(3 \cdot DW + 1) \cdot n_l$$

όπου:

- n_l , k_{ent} , k_{ext} έχουν ήδη οριστεί στο κείμενο
- DW είναι το εύρος της λέξης δεδομένων του επεξεργαστή
- PCW αναφέρεται στο εύρος του απαριθμητή προγράμματος

Πίνακας 3.1: Σύνοψη των επιπρόσθετων πόρων αποθήκευσης που εισάγονται λόγω του ZOLC στον XiRisc.

Unit	Component	Quantity	Num. entries	Bits per entry
Task selection unit	LUT (<i>task_d</i>)	1	$2 \cdot n'_t$	$\log_2(n'_t)$
	LUT (<i>ttset</i>)	1	$2 \cdot n'_t$	1
	LUT (<i>loop_a</i>)	1	$2 \cdot n'_t$	$\log_2(n_l)$
	Output register	1	1	$\log_2(n'_t) + \log_2(n_l) + 1$
	PCent_m register file	k_{ent}	n'_t	PCW
	PCext_m register file	k_{ext}	n'_t	PCW
Loop parameter tables	Register files	3	n_l	DW
Index calculation unit	Mask register	1	1	n_l

3.3.3. Προδιαγραφές σε επίπεδο RTL για τους μηχανισμούς ZOLC

Προκειμένου να βοηθηθεί η εφαρμογή των μηχανισμών ZOLC αναπτύχθηκε από τον ερευνητή Α.Π.Θ. σημειογραφία για την αναπαράσταση του Σχ. 3.7. Μια μεθοδική (formal) περιγραφή των λειτουργιών μιας μονάδας ZOLC σε μικροαρχιτεκτονικό επίπεδο μπορεί να είναι εκμεταλλεύσιμη στο πεδίο της σύνθεσης υψηλού επιπέδου (HLS) για ASIP που στοχεύουν εφαρμογές της περιοχής της ψηφιακής επεξεργασίας σήματος. Ο Πίνακας 3.2 συνοψίζει την σήμανση που χρησιμοποιείται για τα σήματα και τους πόρους αποθήκευσης του ZOLC. Οι απαιτούμενοι πόροι αποθήκευσης είναι:

- Το LUT επιλογής διεργασίας (*ttlut_m*)
- Η συστοιχία καταχωρητών δείκτη (*IXRB*)
- Η συστοιχία καταχωρητών δυναμικής σημαίας (*dynamic flag*) αποτελούμενη από καταχωρήσεις των 2-bit που κωδικοποιούν πληροφορία για τις fwd διεργασίες (*DFRB*)
- Ένας καταχωρητής κατάστασης για τον υπολογισμό δείκτη (*muxsel_m*)
- Οι καταχωρητές παραμέτρων βρόχου (*lparams_m*).
- Αριθμός $NUM_ENTRIES=k_{ent}$ και $NUM_EXITS=k_{ext}$ αρχείων καταχωρητών που απαιτούνται για τον καθορισμό του PC στόχου για τη διάδοχη διεργασία και τις υποψήφιες τιμές PC εξόδου όταν πρέπει να υποστηρίζονται συνθήκες πολλαπλής-εξόδου, αντίστοιχα.

Κατά την αποκωδικοποίηση του PC με το ZOLC σε λειτουργία, εκδίδεται κάποια τιμή *task_d* και η τιμή του PC συγκρίνεται με τις καταχωρήσεις *PCext_m* για το ίδιο *task_d* (για όλες τις εν δυνάμει εξόδους από την τρέχουσα διεργασία). Το διάνυσμα *taskend_t* κωδικοποιεί το αποτέλεσμα αυτής της παράλληλης σύγκρισης, ενώ *zolc_trg* είναι η δυαδική του κωδικοποίηση και το *taskend* είναι ενεργό όταν υπάρχει τουλάχιστον μία ταύτιση με μια

διεύθυνση εξόδου από διεργασία. Σε τέτοια περίπτωση, το ZOLC χρησιμοποιείται για τον καθορισμό της τιμής του ακόλουθου PC και οι απαιτούμενες λειτουργίες διαχωρίζονται στις διαδικασίες LoopParams, IndexCalculation και TaskSelection. Ως αποτέλεσμα αυτών των υπολογισμών προκύπτει μέσα σε ένα κύκλο ρολογιού, μια υπολογισθείσα τιμή PC_zolc η οποία ανατίθεται στον PC και είναι διαθέσιμη κατά τον επόμενο κύκλο.

Το έργο της LoopParams είναι η προσπέλαση των παραμέτρων τρέχοντος βρόχου από τις αντίστοιχες καταχωρήσεις των συστοιχιών καταχωρητών, οι οποίες διευθυνσιοδοτούνται από το loop_a. Για το σκοπό αυτό, ο καταχωρητής κατάστασης muxsel_m σημειώνει το για ποιο βρόχο πρέπει να αρχικοποιηθεί ο δείκτης βρόχου κατά την επόμενη ανανέωση δείκτη. Το σήμα loopend, που επίσης υπολογίζεται από τη μονάδα υπολογισμού δείκτη, ανακοινώνει την τελική επανάληψη ενός βρόχου. Η TaskSelection περιγράφει τον ουσιαστικό μηχανισμό για τη μεταγωγή διεργασιών. Για τις bwd και τις τετριμμένες (αυτές που δεν τερματίζονται υπό συνθήκη) fwd διεργασίες, η διεύθυνση διεργασίας (task_a) για τη διάδοχη διεργασία σχηματίζεται με τη συνένωση (concatenation) των σημάτων task_d και loopend. Για την έξοδο από fwd διεργασίες με δυναμικές διακλαδώσεις απαιτείται επιπρόσθετη πληροφορία για τον στόχο της διακλάδωσης, η οποία κωδικοποιείται στον καταχωρητή κατάστασης DFRB. Στην παρουσία ενός ενεργού taskend, η πληροφορία για τη διάδοχη διεργασία διαβάζεται από το LUT επιλογής διεργασίας. Τα αρχεία καταχωρητών PCent_m και PCext_m διευθυνσιοδοτούνται από το task_d. Οι συγκεκριμένες καταχωρήσεις για την zolc_trg διεύθυνση στήλης από τα PCent_m και PCext_m παρέχουν το κατάλληλο PC_zolc και τον PC εξόδου διεργασίας για την διάδοχη διεργασία, αντίστοιχα. Αυτή η προσέγγιση απαιτεί ότι το LUT επιλογής διεργασίας μπορεί να αναγνωστεί ασύγχρονα.

```

// PC decoding operations
PCDecoding()
begin
  for i in 0..NUM_EXITS-1 do
    if PC equals PCext_m[task_d].i then
      taskend_l[i] := 1
    else
      taskend_l[i] := 0
    end if
  end for

  if not(loopend) and not(ttsel) then
    IXRB_m[loop_a] := index_l
  elsif loopend and not(ttsel) then
    IXRB_m[loop_a] := initial
  end if

  if taskend and not(ttsel) and not(loopend) and
  not(muxsel_m[loop_a]) then
    muxsel_m[loop_a] := 1
  elsif taskend and not(ttsel) and loopend then
    muxsel_m[loop_a] := 0
  end if
end if

end

// task switching mechanism
TaskSelection()
begin
  if not(ttsel) then
    task_a := task_d concat loopend
  else
    switch on DFRB[task_d]
      when 0: task_a := task_d concat loopend
      when 1: task_a := task_d concat 0
      when 2: task_a := task_d concat 1
    end switch
  end if

  if taskend then
    task_d := tflur_m[task_a].TASK_DATA
    ttsel := tflur_m[task_a].TTSEL
    loop_a := tflur_m[task_a].LOOP_A
  end if

  PC_zolc := PCent_m[task_d].zolc_arg
  PC_task_ext := PCext_m[task_d].zolc_arg
end
    
```

Σχήμα 3.7: Ψευδοκώδικας για τις λειτουργίες ZOLC σε μικροαρχιτεκτονικό επίπεδο.

Πίνακας 3.2: Σήμανση που χρησιμοποιείται στον ψευδοκώδικα του Σχήματος 3.7.

Name	Description
{name}_a	Address signal
{name}_d	Data signal
{name}[ent—ext]	Referring to loop entry/exit
{name}_m	Register (file)/memory resource
{name}_l	Temporary
*.{name in caps}	Field name
*.{name in lowercase}	Column select for 2D arrays

3.4. Διεξοδική μελέτη των απαιτήσεων για την υποστήριξη του ZOLC στο υλικό και στο λογισμικό

3.4.1. Αυτοματοποιώντας τις βελτιστοποιήσεις ZOLC εντός πλαισίου εργασίας μεταγλωττιστή-βελτιστοποιητή επιπέδου συμβολομεταφραστή

Για να υποστηριχθεί επαρκώς ενδεχόμενη χρήση της μονάδας ZOLC, απαιτούνται κατάλληλα εργαλεία ανάπτυξης εφαρμογών (software development tools) ώστε να διευκολυνθεί η γένεση κώδικα από γλώσσες προγραμματισμού υψηλού επιπέδου (C, C++). Για τις καλύτερες, το δυνατόν, επιδόσεις, ένα επιπρόσθετο πέρασμα μεταγλωττιστή θα πρέπει να εκτελείται για την εκμετάλλευση του ZOLC, προηγούμενο της εκχώρησης καταχωρητών και αμέσως μετά την οικοδόμηση του βελτιστοποιημένου CFG. Σκοπός του είναι η εκπομπή κώδικα αρχικοποίησης του ZOLC και η εισαγωγή των εντολών μετακίνησης που απαιτούνται για τους βρόχους με δυναμικώς καθοριζόμενες οριακές τιμές.

Σχετικά με τον XiRisc, η αλυσίδα εργαλείων ανάπτυξης βασίζεται στον μεταγλωττιστή gcc. Ο απαιτούμενος κόπος για να καταστρωθεί και να εισαχθεί ένα εσωτερικό πέρασμα στον gcc μπορεί να είναι υπερβολικά μεγάλος μιας και ο gcc έχει σχεδιασθεί περισσότερο για την προσθήκη νέων αρχιτεκτονικών μηχανής και όχι για ευκολία στην ανάπτυξη νέων περασμάτων μεταγλωττιστή όπως για τον μετασχηματισμό σε επίπεδο IR (σημ.: η κατάσταση βελτιώνεται σταδιακά από την έκδοση 4.0.0 με την υποδομή Tree-SSA). Το πρόβλημα μπορεί να παρακαμφθεί με την μετεπεξεργασία (post-processing) του παραγόμενου κώδικα συμβολομεταφραστή (π.χ. για τον XiRisc) και αυτή είναι η προσέγγιση που τέθηκε σε χρήση ώστε να ληφθούν τα αποτελέσματα της ενότητας 3.5.2.

Προκειμένου να αποδείξουμε την εφαρμοσιμότητα της παραπάνω αρχής για την υποστήριξη του ZOLC σε ένα αυτοματοποιημένο πλαίσιο εργασίας μεταγλωττιστή, αναπτύχθηκε πέρασμα ανάλυσης και επισήμανσης (analysis and markup) για τον Machine-SUIF [MachSUIF]. Το πέρασμα αυτό εργάζεται στο επίπεδο CFG και χρησιμοποιείται ώστε να περαστούν μετα-εντολές και επιπρόσθετη πληροφορία σε ένα πέρασμα μετασχηματισμού για το εργαλείο μετασχηματισμού κώδικα συμβολομεταφραστή SALTO [SALTO],[Roh96] ώστε την οριστικοποίηση της ακολουθίας εντολών αρχικοποίησης του ZOLC. Η στοχευόμενη αρχιτεκτονική για αυτό το πλαίσιο εργασίας επίδειξης ονομάζεται «πραγματική μηχανή SUIF» (SUIFrm) για την οποία ο ερευνητής Α.Π.Θ. συνέταξε ένα πειραματικό backend για τον Machine-SUIF και ένα σύνολο από προσομοιωτές ακρίβειας εντολής και κύκλου για την υποδομή ArchC [ArchC]. Η SUIFrm δανείζεται πολλά στοιχεία από τη συμβολική αρχιτεκτονική-στόχο SUIFvm (IR του Machine-SUIF). Οι κύριες προσθήκες του SUIFrm backend στο SUIFvm είναι:

- Ομογενής αρχιτεκτονική καταχωρητών (ο αριθμός καταχωρητών ρυθμίζεται σε 12, 16, 32, 64).
- Κατάλληλη διερμηνεία των γενικών εντολών μετατροπής τύπου (cvt) σε μορφές επέκτασης μηδενικού (zxt), επέκτασης προσήμου (sxt) και αποκοπής (trunc).
- Πέρασμα των ορισμάτων συναρτήσεων μέσα από εξειδικευμένους καταχωρητές ορισμάτων.

Κάποια χαρακτηριστικά που λείπουν από τη SUIFrm αρχιτεκτονική επηρεάζουν τη δυνατότητα του περάσματος στη στόχευση εφαρμογών γενικού σκοπού: α) έλλειψη υποστήριξης πράξεων κινητής υποδιαστολής, και β) δομές (structs), ενώσεις (unions) και αναδρομικές συναρτήσεις όπως ορίζονται για την C δεν υποστηρίζονται. Παρ' όλα αυτά, το backend για την SUIFrm μπορεί να χρησιμοποιηθεί για όλες τις εφαρμογές δοκιμής της ενότητας 3.5.2. Επίσης, τα αντίστοιχα SUIFrm backend (με και χωρίς την υποστήριξη ZOLC) αναπτύχθηκαν και για την υποδομή SALTO.

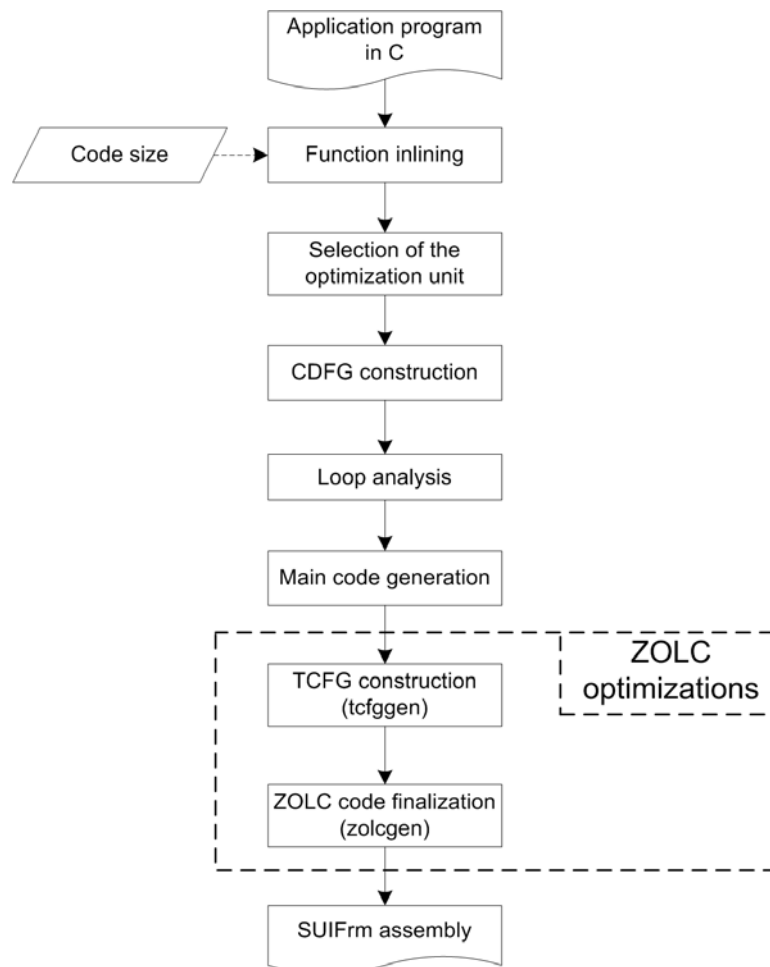
Το Σχ. 3.8 απεικονίζει την διαδικασία γένεσης κώδικα όπως αυτή ενσωματώνεται στην ροή μεταγλώττισης του ερευνητή Α.Π.Θ., η οποία αποτελείται από ένα SUIF frontend, το

Machine-SUIF backend που αναφέρθηκε, συμπληρωμένα από τα περάσματα για τις βελτιστοποιήσεις ZOLC τα οποία γράφτηκαν για τα Machine-SUIF και SALTO σύμφωνα με το API που ακολουθούν. Η λειτουργία μπορεί να διαχωριστεί στα ακόλουθα στάδια:

- Διαγράμμιση συνάρτησης
 - Γίνεται επιλεκτική διαγράμμιση συνάρτησης ώστε να εκθεθούν όσο το δυνατόν περισσότερα βασικά μπλοκ στην συνάρτηση του προγράμματος εφαρμογής που βρίσκεται στο ανώτερο επίπεδο, εντός κάποιων ορίων για το μέγεθος κώδικα.
- Επιλογή της μονάδας βελτιστοποίησης
 - Ως συνέπεια της λειτουργίας πάνω σε CFG και όχι σε μορφές IR που καταγράφουν σε μια οντότητα τη συμπεριφορά όλης της στοχευόμενης εφαρμογής (τέτοιιοι είναι γράφοι εξάρτησης προγράμματος – PDGs), είναι αναγκαίο να επιλέγεται μία μόνη συνάρτηση για την εφαρμογή των βελτιστοποιήσεων ZOLC. Η μονάδα βελτιστοποίησης επιλέγεται με βάση το προφίλ συχνοτήτων εκτέλεσης βασικών μπλοκ της αναλυόμενης εφαρμογής.
- Οικοδόμηση του CDFG
 - Κατασκευάζεται η συλλογή των CDFG ολόκληρης της εφαρμογής. Υποτίθεται εδώ ότι μπορεί να εφαρμοστεί δρομολόγηση εντολών τουλάχιστον σε τοπικό επίπεδο (εντός των βασικών μπλοκ).
- Ανάλυση βρόχων
 - Το βήμα αυτό έχει στόχο την αναγνώριση των δηλώσεων βρόχων της εφαρμογής, είτε με την τεχνική της ανάλυσης φυσικών βρόχων [Aho86] που υποστηρίζεται στον Machine-SUIF είτε κατά προτίμηση από περισσότερο εξεζητημένες μεθόδους για την ανίχνευση βρόχων πολλαπλών εισόδων. Η παραγόμενη αναφορά φυσικής ανάλυσης βρόχων περιέχει το βάθος εμφώλευσης βρόχων (loop nesting depth) και τρεις επιπρόσθετες σημαίες για τον καθορισμό των: α) αν ένας βρόχος αρχίζει από τον συγκεκριμένο κόμβο (begin_node), β) αν ένας βρόχος τερματίζεται στον συγκεκριμένο κόμβο (end_node), και γ) αν είναι εφικτή η έξοδος από το βρόχο από τον συγκεκριμένο κόμβο (exit_node). Τεχνικές γενικευμένης ανάλυσης βρόχων που ανιχνεύουν βρόχους με πολλαπλές εισόδους/εξόδους από/σε μη-ακολουθιακά βασικά μπλοκ, θα απαιτούσαν επιπλέον, τη γνώση του σε ποιο βασικό μπλοκ μεταφέρεται ο έλεγχος από το προγενέστερο βασικό μπλοκ εξόδου από βρόχο. Ελάχιστες πειραματικές υποδομές μεταγλωττιστή υποστηρίζουν την άμεση ανίχνευση βρόχων πολλαπλών εισόδων [Bro04].
- Κύρια γένεση κώδικα
 - Στη συνέχεια χρησιμοποιούνται, ο επιλογέας εντολών (πέρασμα 'do_gen') και ο εκχωρητής καταχωρητών ('do_raga'). Ο επιλογέας εντολών του Machine-SUIF αναλαμβάνει την αποτύπωση του SUIFvm IR σε SUIFrm κώδικα και απλώς υλοποιεί έναν απλοϊκό μεταφραστή από SUIFvm σε SUIFrm χωρίς χρήση κάποιας τεχνικής για τη παραγωγή των βέλτιστων μοτίβων όπως είναι η BURS [IBURG]. Ο εκχωρητής καταχωρητών είναι σαφώς πιο εκλεπτυσμένος και υλοποιεί τη γνωστή μέθοδο της επαναληπτικής συνάσπισης καταχωρητών (iterated register coalescing) [Geo96]. Με το πέρας του σταδίου αυτού μπορεί να εκπεμφθεί SUIFrm κώδικας συμβολομεταφραστή.
- Οικοδόμηση TCFG (πέρασμα tcfggen)
 - Με βάση τα αποτελέσματα της ανάλυσης βρόχων αποτυπώνεται η ροή ελέγχου της εφαρμογής στο TCFG της. Επιπλέον, το στάδιο αυτό προωθεί τη

στατική πληροφορία των βρόχων του αλγορίθμου (που υλοποιεί η εφαρμογή) σε επόμενο στάδιο, αυτό δε γίνεται με χρήση τεσσάρων διαφορετικών τύπων μετα-εντολών. Μια LDST εντολή ενσωματώνει όλη την απαιτούμενη πληροφορία για τη δημιουργία μιας καταχώρησης στο LUT επιλογής διεργασίας: για μια δοσμένη κωδικοποίηση διεργασίας (*current-task_d*), δίνονται η διάδοχη διεργασία (*next-task_d*), ο τύπος της (*next-ttsel*) και η διεύθυνση βρόχου (*next-loop_a*). Οι εντολές DPTI σημειώνουν το πρώτο και τελευταίο βασικό μπλοκ μιας διεργασίας, ενώ μια εντολή LOOP παρέχει μια δοσμένη διεύθυνση βρόχου, τον καταχωρητή δείκτη βρόχου, και τις παραμέτρους βρόχου. Μια OVERHEAD σημειώνει το αν η επόμενη ψευδο-εντολή της θα πρέπει να διατηρηθεί στο πρόγραμμα (προεπιλογή), να αντικατασταθεί από NOP, ή να απομακρυνθεί εντελώς.

- Οριστικοποίηση του κώδικα ZOLC (πέραςμα *zolcgen*)
 - Το πέραςμα *zolcgen* για το SALTO παράγει τον κώδικα αρχικοποίησης του ZOLC ο οποίος πρέπει να εισαχθεί σε ένα προγενέστερο βασικό μπλοκ της δομής βρόχων για την ανανέωση των πόρων αποθήκευσης του ZOLC και αυτό είναι συνήθως το πρώτο βασικό μπλοκ στην στοχευόμενη συνάρτηση. Η διαδικασία αυτή περιλαμβάνει τα ακόλουθα:
 - Μετατροπή των ψευδοεντολών LOOP σε μια ακολουθία LDSA-LDSI-LDSS-LDSF εντολών (και την επανατοποθέτησή τους). Οι εντολές LDS[|S|F] ερμηνεύονται στον Πίνακα 3.3 ενώ η LDSA (Set index register alias) αντιστοιχίζει τον καταχωρητή γενικού σκοπού που χρησιμοποιείται για την δεικτοδότηση σε ένα δεδομένο βρόχο με την τιμή *loop_a*. Οι LDSA μπορούν να χρησιμοποιηθούν για τη στατική μετονομασία σε περίπτωση αρχιτεκτονικής με αφιερωμένους καταχωρητές δείκτη βρόχου. Για ομογενή αρχιτεκτονική καταχωρητών αυτή η πληροφορία θα αποτυπωνόταν σε ένα μικρό LUT για την παροχή της συσχέτισης των καταχωρητών δεικτοδότησης με τις αντίστοιχες τους διευθύνσεις βρόχων.
 - Γίνεται διαχείριση των εντολών επιβάρυνσης και οι αντίστοιχοι ενδείκτες απομακρύνονται.
 - Οι διευθύνσεις PC εισόδου/εξόδου από διεργασία υπολογίζονται και δημιουργούνται οι αντίστοιχες εντολές LDS[N|X] (Set a PC entry/exit address).



Σχήμα 3.8: Τμήμα ροής μεταγλώττισης που υποστηρίζει την αυτοματοποίηση της γένεσης κώδικα με ενδεχόμενη χρήση των μηχανισμών ZOLC.

Πίνακας 3.3: Εντολές επέκτασης για την υποστήριξη του ZOLC στον επεξεργαστή XiRisc.

Mnemonic	Description
LDSL	Set task transition LUT entry
LDSNi	Set PC entry address for $k_{ent} = i$
LDSXi	Set PC exit address for $k_{ext} = i$
LDSM	Set mask register for $mutexelm$ and index calculation ALU initialization
LDS[I S F]	Set initial/step/final loop parameter
MOVG2[I S F]	Move a general-purpose register to an initial/step/final loop parameter register
MOV[I S F]2G	Move an initial/step/final loop parameter register to a general-purpose register

3.4.2. Αναλύοντας διεξοδικά τις επιπλοκές του ZOLC στο υλικό και στο λογισμικό των ενσωματωμένων επεξεργαστών

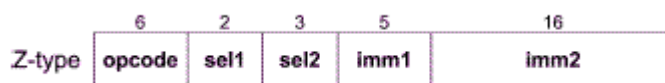
Η τεχνική ZOLC συνιστά βελτιστοποίηση για το διάδρομο ελέγχου και είναι εφαρμόσιμη (όπως περιγράφεται εδώ) στους ενσωματωμένους επεξεργαστές με ολοκλήρωση-σε-σειρά (in-order completion). Συγκεκριμένα ζητήματα που αφορούν τη χρήση του ZOLC επισημαίνονται στις αμέσως επόμενες παραγράφους.

- Ενσωμάτωση του ZOLC σε VLIW επεξεργαστές

- Θεωρητικά, η τεχνική ZOLC θα μπορούσε να εφαρμοστεί σε VLIW επεξεργαστές, με τη διαφορά της κατά πολύ αυξημένης επιβάρυνσης για την παρακολούθηση της πλεονάζουσας κατάστασης λόγω της επίδρασης της λογισμικής διοχέτευσης (software pipelining), η οποία υπονοεί ότι, σε κάθε στιγμή, είναι ενεργό ένα περιεχόμενο πολλαπλών δεικτών (δηλ. είναι σε εξέλιξη πολλαπλές επαναλήψεις βρόχων στο ίδιο βήμα ελέγχου). Όπως αναφέρεται στο [Eyr00] οι σύγχρονοι VLIW DSP τείνουν να παραβλέπουν τα ZOLC χαρακτηριστικά. Για παράδειγμα, η οικογένεια TMS320C62xx (με 8 θυρίδες έκδοσης εντολών) δεν υποστηρίζει εντολές τύπου ZOLC αλλά απαιτεί από τον επεξεργαστή να εκτελέσει με κλασσικές εντολές τις λειτουργίες επιβάρυνσης βρόχου. Αυτό είναι αποδεκτό στην περίπτωση των VLIW για αυτοί είναι ικανοί να εκτελούν σημαντικό αριθμό εντολών σε παραλληλία. Οι λειτουργίες που χρειάζονται για την διαχείριση του βρόχου μπορούν να εκτελεστούν παράλληλα ως προς άλλους αριθμητικούς υπολογισμούς, υποθέτοντας ότι ο μεταγλωττιστής για τον επεξεργαστή μπορεί να καταλήξει σε πλεονεκτική δρομολόγηση εντολών. Για μια αισιόδοξη περίπτωση δρομολόγησης, θα μπορούσαν να επιτευχθούν οι ίδιες επιδόσεις (όσον αφορά την επιβάρυνση βρόχων) με τους επεξεργαστές που διαθέτουν τέτοια υποδομή στο υλικό.
- Ενσωμάτωση της εξαγωγής TCFG για την γένεση κώδικα ZOLC στους υπάρχοντες μεταγλωττιστές
 - Ένα έμφυτο χαρακτηριστικό των τύπου-CDFG IR μεταγλωττιστών (όπως είναι κάποια IR SSA μορφής) είναι η αναπαράσταση κάθε συνάρτησης από ένα CFG κάτι που περιπλέκει την εξαγωγή TCFG και που άμεσα επηρεάζει την ποιότητα της παραγωγής κώδικα ZOLC. Η έκταση στην οποία πρέπει να εφαρμόζονται βελτιστοποιήσεις όπως είναι η διαγράμμιση συνάρτησης είναι ένα από τα βασικά προβλήματα αυτής της προσέγγισης. Από την άλλη πλευρά, η αναπαράσταση PDG [Fer87] επιτρέπει αλγορίθμους ανάλυσης εξαρτήσεων ελέγχου για πολλαπλές συναρτήσεις κάτι που είναι περισσότερο συμβατό με τις προϋποθέσεις για την εξαγωγή ενός TCFG σε εμβέλεια προγράμματος. Μια βιβλιοθήκη με τα αντίστοιχα περάσματα για την εξαγωγή μιας μορφής γράφου εξάρτησης ελέγχου (CCDG) για τον Machine-SUIF έχει αναφερθεί [She04] και βρίσκεται υπό εξέταση για μέλλουσα επέκταση της γεννήτριας κώδικα ZOLC του ερευνητή Α.Π.Θ..
- Επέκταση ρεπερτορίου εντολών για την υποστήριξη ZOLC
 - Για την αρχικοποίηση του ZOLC από το λογισμικό, απαιτείται ένα μικρό σύνολο εντολών επέκτασης για την εκτέλεση εξειδικευμένων εντολών αποθήκευσης στο επιπρόσθετο υλικό του ZOLC. Έτσι, για την προσέγγιση που ακολουθήθηκε για την ενσωμάτωση του ZOLC στον XiRisc, τρεις νέες εντολές (LDS, MOVG2X, MOVX2G) σχεδιάστηκαν με την διαμόρφωση του Σχ. 3.9. Όλες οι εντολές που προστέθηκαν συνοψίζονται στον Πίνακα 3.3.
- Επιλογή αρχιτεκτονικής καταχωρητή (ξεχωριστή συστοιχία καταχωρητών δείκτη ή αφιέρωση νέας κλάσης καταχωρητών για τη δεικτοδότηση)
 - Παραδοσιακά, οι DSP αρχιτεκτονικές χρησιμοποιούν συστοιχίες καταχωρητών δείκτη το οποίο επιτρέπει περισσότερο αποδοτική χρησιμοποίηση σύνθετων τρόπων διευθυνσιοδότησης. Όμως, η ευκολία των εκχώρησης καταχωρητών για ομογενείς αρχιτεκτονικές καταχωρητών στους πρώτους RISC που εμφανίστηκαν κατέστησε ασύμφορη τη χρήση καταμεμημένων πόρων αποθήκευσης για τις λειτουργίες καταχώρησης. Για το λόγο αυτό, η προσέγγιση της συστοιχίας καταχωρητών δείκτη είναι βιώσιμη μόνο σε νέους ASIP χωρίς απαίτηση συμβατότητας (δυναμικής ή κώδικα συμβολομεταφραστή) με παλαιότερες αρχιτεκτονικές. Παρ' όλα αυτά, η έτερη

τεχνική (νέα κλάση καταχωρητών δείκτη που βρίσκονται στο γενικό αρχείο καταχωρητών) μπορεί να εφαρμοστεί σε οικογένειες ενσωματωμένων επεξεργαστών, παρέχοντας συμβατότητα σε επίπεδο συμβολομεταφραστή η οποία διατηρείται με μετασχηματισμούς σε αυτό το επίπεδο. Καθώς συνήθως οι τιμές δείκτη χρειάζονται μειωμένο εύρος bit, η εκτιμώμενη επιφάνεια υλικού αλλά και η κατανάλωση ισχύος μειώνονται σημαντικά. Σε ένα τέτοιο σχήμα, μπορούν να χρησιμοποιηθούν καταχωρητές περιορισμένου εύρους bit σε μια ομογενή αρχιτεκτονική με τον τρόπο που αυτό γίνεται πράξη στα ασυμμετρικά αρχεία καταχωρητών (asymmetrically-ported register files [Agg03]) με σημαντικά οφέλη ιδιαίτερα στην κατανάλωση ισχύος.

- Μεταγωγή διεργασιών πάνω σε εντολές πολλαπλών κύκλων ή εντολές με χρονισμό εξαρτημένο από τα δεδομένα
 - ο Η επιλογή της διάδοξης διεργασίας πραγματοποιείται με δυναμικό τρόπο κατά την ανάκληση εντολής. Οι μικρολειτουργίες όμως με τις οποίες περιγράφηκε αυτό στην ενότητα 3.3 (κυρίως στην υποενότητα 3.3.3), επαρκούν για τις απλούστερες περιπτώσεις λειτουργικών μονάδων του επεξεργαστή, καθώς υποθέτουν ότι η τελευταία εντολή σε ένα DPT είναι ενός κύκλου και ο επεξεργαστής υποστηρίζει μόνο ολοκλήρωση-σε-σειρά. Για εντολές πολλαπλών κύκλων γνωστού χρονισμού, η διαδικασία της απόφασης (τερματισμού της τρέχουσας διεργασίας) πρέπει να φρουρείται μέχρι τον κατάλληλο αριθμό κύκλων πριν την αποδέσμευση της εκτελούμενης εντολής. Για παράδειγμα, για έναν πολλαπλασιασμό 4-κύκλων, η δυναμική αυτή απόφαση πρέπει να ενεργοποιείται 3 κύκλους μετά την ανάκληση της εντολής από τη μνήμη προγράμματος. Εντολές για τις οποίες ο χρονισμός είναι άγνωστος κατά το χρόνο μεταγλώττισης (π.χ. ο πολλαπλασιασμός στον ARM7TDMI απαιτεί 2 ως 5 κύκλους μηχανής στο στάδιο εκτέλεσης ανάλογα με τις τιμές των έντελων), μειώνουν τις επιδόσεις του ZOLC και, το δυνατόν, θα πρέπει να δρομολογούνται σε κατάλληλες χρονοθυρίδες από τον μεταγλωττιστή.



Σχήμα 3.9: Προτεινόμενη διαμόρφωση εντολής για την αρχικοποίηση του ZOLC. Υποτίθεται ότι $kent = kext = 4$.

3.5. Εκτίμηση επιδόσεων για RISC μικροεπεξεργαστές που διαθέτουν μηχανισμούς ZOLC

3.5.1. Απαιτήσεις σε υλικό για την υλοποίηση μηχανισμών ZOLC σε ενσωματωμένους επεξεργαστές

Για τον σκοπό της εκτίμησης επιδόσεων, υλοποιήθηκαν τρεις διαφορετικές εκδοχές μιας μηχανής ZOLC. Ως ZOLCfull αναφέρεται η εκδοχή πλήρων δυνατοτήτων που υποστηρίζει 32 καταχωρήσεις για μεταγωγές διεργασιών (που αντιστοιχεί σε 16 DPT μιας εισόδου) και είναι ικανή να καταγράψει το σύνολο της πληροφορίας ZOLC για αυθαίρετες δομές με μέγιστο 8 βρόχων και 4 εισόδους/εξόδους ανά διεργασία. Η εκδοχή ZOLClite διαφοροποιείται από την ZOLCfull καθώς ελλείπεται της υποστήριξης βρόχων πολλαπλών εισόδων/εξόδων. Η uZOLC είναι η μινιμαλιστική εκδοχή μιας μηχανής ZOLC που είναι χρησιμοποιήσιμη μόνο για ένα βρόχο κάθε φορά. Και οι τρεις διαφορετικές εκδοχές ενσωματώθηκαν στην VHDL περιγραφή του XiRisc (του οποίου η βασική υλοποίηση ονομάζεται XRdefault). Οι τέσσερις αυτές

υλοποιήσεις του XiRisc συντέθηκαν σε διεργασία standard cell STM 0.13μm. Δοθέντος ότι τα αποτελέσματα ελήφθησαν πριν το στάδιο τοποθέτησης-και-διασύνδεσης (place-and-route), η μέγιστη συχνότητα ρολογιού για τον επεξεργαστή δεν βρέθηκε να επηρεάζεται (170MHz σε όλες τις περιπτώσεις). Αυτό οφείλεται στο γεγονός ότι ως κρίσιμο μονοπάτι παρέμεινε αυτό που διέρχεται από τον πολλαπλασιαστή του XiRisc (1/1 throughput/latency). Στην πραγματικότητα, η εκτιμώμενη μέγιστη συχνότητα ρολογιού από το εργαλείο σύνθεσης για την εκδοχή ZOLCfull ήταν ακόμη υψηλότερη από την εκδοχή XRdefault, αυτό όμως φαίνεται ότι οφείλεται στους υπολογισμούς για τα δικτυώματα RC που μοντελοποιούν τις διασυνδέσεις των βασικών κελιών.

Η επιβάρυνση σε επιφάνεια συνδυαστικού υλικού μετρήθηκε από τις λίγες εκατοντάδες πύλες (298 για τον uZOLC) στις λίγες χιλιάδες πύλες (4428 για τον ZOLCfull και 4056 για τον ZOLClite) ενώ ο μη τροποποιημένος XiRisc συντίθεται σε 21309 ισοδύναμες NAND πύλες (περιλαμβάνοντας και τους πόρους αποθήκευσης: αρχείο καταχωρητών γενικού σκοπού και καταχωρητές διοχέτευσης) παρέμεινε να είναι αυτό που διέρχεται από τον πολλαπλασιαστή του XiRisc (1/1 latency/throughput). Για τις προεπιλεγμένες ρυθμίσεις του XiRisc (PCW=16, DW=32) εύκολα μπορούν να εξαχθούν οι απαιτήσεις σε πόρους αποθήκευσης για τις τρεις εκδοχές της μηχανής ZOLC. Όπως προκύπτει από την αναλυτική έκφραση της ενότητας 3.3.2, χρειάζονται 171, 1552, και 3088 bit αποθήκευσης για τις εκδοχές uZOLC, ZOLClite και ZOLCfull, αντίστοιχα. Αν και κάτι τέτοιο δεν υλοποιήθηκε, ο αριθμός των bit αποθήκευσης θα μειωνόταν σημαντικά αν χρησιμοποιούνταν κυκλώματα επέκτασης μεγέθους για τις (σχετικές) αποστάσεις διακλάδωσης που αποθηκεύονταν στους πίνακες PC εισόδου και εξόδου διεργασίας (οι αποθηκευμένες τιμές πρέπει να είναι στα 8-bit, επεκτάσιμες σε PCW). Αν αυτό είχε εφαρμοστεί, θα είχε εξοικονομηθεί το 22.8% των bit αποθήκευσης.

3.5.2. Επιδόσεις σε κύκλους μηχανής για έναν MIPS-I επεξεργαστή με αρχιτεκτονικές τροποποιήσεις κατά ZOLC

Καθώς ο XiRisc διαθέτει ρεπερτόριο εντολών συμβατό με την αρχιτεκτονική MIPS-I, ο ερευνητής τροποποίησε κατάλληλα ένα ήδη υπάρχον μοντέλο (σε ArchC) για επεξεργαστή MIPS R3000 και προσέθεσε στην αντίστοιχη περιγραφή τη συμπεριφορά των λειτουργιών ZOLC (βασιζόμενος στον ψευδοκώδικα της ενότητας 3.3.3) ώστε την δημιουργία ενός προσομοιωτή ακρίβειας κύκλου για τον XiRisc.

Για την εκτίμηση επιδόσεων συγκεντρώθηκαν δύο σύνολα από εφαρμογές δοκιμής, οι οποίες περιγράφονται λεπτομερώς στον Πίνακα 3.4 μαζί με τον αριθμό δυναμικών εντολών τους (για την εκτέλεση χωρίς χρήση ZOLC). Το πρώτο σύνολο αποτελείται 10 εφαρμογές-πυρήνες με εντατική επεξεργασία δεδομένων σε δομές βρόχων, κοινές σε μεγάλο αριθμό εφαρμογών, κάθε μία δε από τις οποίες αποτελείται ένα αρχείο σε ANSI C με ενσωματωμένα, στον πηγαίο κώδικα, τα απαραίτητα δεδομένα εισόδου. Η σουίτα αυτή των εφαρμογίδων αναφέρεται και ως ZOLCbench. Το δεύτερο σύνολο αποτελείται από 10 εφαρμογές της δημοφιλούς σουίτας εφαρμογών MiBench [Gut01], για τις οποίες χρησιμοποιήθηκαν οι λεγόμενες «μικρές» είσοδοι (small inputs). Όλες οι εφαρμογές δοκιμής μεταγλωττίστηκαν από τον gcc με τις επιλογές "-O3 -mips1" με εξαίρεση τις εφαρμογές επεξεργασίας εικόνας susan-* για τις οποίες ενεργοποιήθηκε η προσομοίωση των πράξεων κινητής υποδιαστολής από ρουτίνες που περιλαμβάνουν αποκλειστικά πράξεις ακεραίων (-msoft-float [SoftFloat]) καθώς αυτό ήταν απαραίτητο. Για τις μετρήσεις χρησιμοποιήθηκαν οι εκδοχές uZOLC και ZOLClite. Για την πρώτη περίπτωση, εφαρμόστηκε (χειρωνακτικά) μετασχηματισμός κατανομής βρόχου (loop distribution) ώστε τον κατάλληλο διαχωρισμό σε μικρότερες δομές βρόχων για τις εφαρμογές fsme_dr (20 βρόχοι) και gcdct (14 βρόχοι).

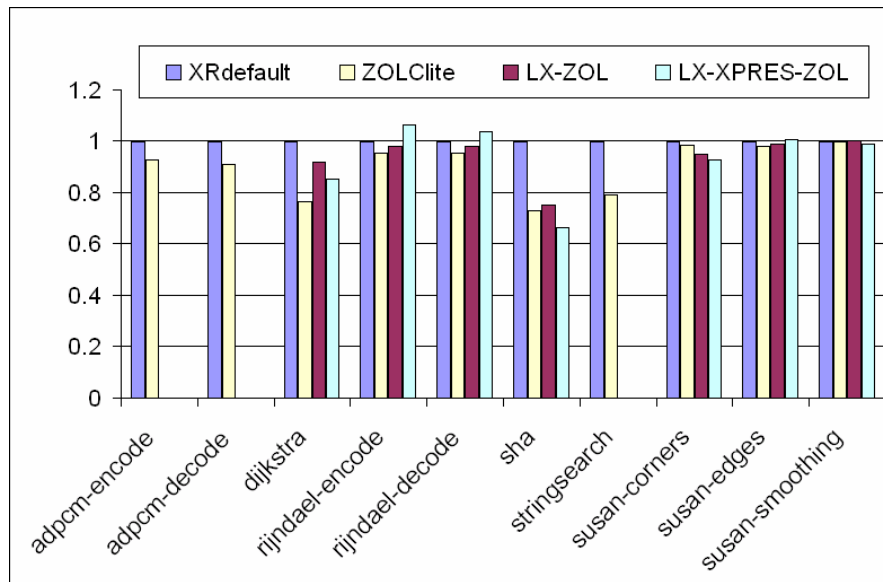
Πίνακας 3.4: Σύνοψη των εφαρμογών δοκιμής που εξετάστηκαν.

Benchmark	Description	Dynamic instructions
Data-intensive kernels		
<i>fsme</i>	Full-search motion estimation	22,030,675
<i>issme</i>	Three-step logarithmic search motion estimation	2,101,005
<i>fsme_dr</i>	Full-search motion estimation with data-reuse transformations	13,052,691
<i>rcdct</i>	Row-column decomposition forward DCT	836,568
<i>matmult</i>	Blocked matrix multiplication	374,355
<i>edgedet</i>	Gradient-based edge detection	91,091
<i>mc</i>	Motion compensation	113,382
<i>kmpskip</i>	Knuth-Morris-Pratt string matching algorithm	30,437
<i>lcs</i>	Longest common substring search algorithm	1,313,236
<i>wsqc</i>	Wavelet-Scalar Quantization image compression	479,596
Embedded applications		
<i>adpcm-encode</i>	Adaptive Differential Pulse Code Modulation (ADPCM) encoder	34,628,152
<i>adpcm-decode</i>	Adaptive Differential Pulse Code Modulation (ADPCM) decoder	27,256,674
<i>dijkstra</i>	Shortest path calculation between node pairs with Dijkstra's algorithm	59,354,952
<i>rijndael-encode</i>	Advanced Encryption Standard encoder	33,697,370
<i>rijndael-decode</i>	Advanced Encryption Standard decoder	34,666,810
<i>sha</i>	Secure Hash Algorithm producing an 160-bit message digest for a given input	13,036,406
<i>stringsearch</i>	Case-insensitive string matching	279,725
<i>susan-corners</i>	Corner detection with the SUSAN image processing package	3,469,990
<i>susan-edges</i>	Edge detection with the SUSAN image processing package	6,898,760
<i>susan-smoothing</i>	Structure-preserving image noise reduction with the SUSAN image processing package	35,331,316

Οι σχετικές μετρήσεις για τους κύκλους εκτέλεσης των εφαρμογών δοκιμής δίνονται στο Σχ. 3.10. Από τις μετρήσεις προκύπτει ότι η υλοποίηση ZOLClite είναι υπεύθυνη για μέση μείωση του αριθμού κύκλων εκτέλεσης κατά 25.5% και 10% για τους πυρήνες (kernels) και τις εφαρμογές MiBench, αντίστοιχα. Για ορισμένους αλγορίθμους (*rcdct*, *matmult*) η βελτίωση στις επιδόσεις ξεπερνά το 40% λόγω της χρήσης του ZOLC. Επίσης, συμπεραίνεται ότι η εκδοχή uZOLC αποτελεί έναν ικανοποιητικό συμβιβασμό για τις περισσότερες εφαρμογές-πυρήνες, ιδιαίτερα για αυτές που χαρακτηρίζονται από πλήρως φωλιασμένους βρόχους, όπου ένας μόνο εσώτερος βρόχος αποτελεί αντικείμενο ενδιαφέροντος για βελτιστοποίηση κατά ZOLC, καθώς επιφέρει μέση επιτάχυνση κατά 17.5%.



(α)



(β)

Σχήμα 3.10: Αποτελέσματα επιδόσεων κύκλων εκτέλεσης για τις εξετασθείσες εφαρμογές δοκιμής. (α) Εφαρμογές-πυρήνες. (β) Εφαρμογές δοκιμής που επιλέχθηκαν από την σουίτα MiBench.

Προκειμένου την ακόμη καλύτερη υποστήριξη της επάρκειας του ZOLC, ελήφθησαν μετρήσεις για έναν κορυφαίο προσδιορισμό RISC επεξεργαστή της αγοράς ο οποίος διαθέτει βελτιστοποιήσεις ZOL για απλούς βρόχους, ονόματι LX 1.0 ο οποίος ανήκει στην οικογένεια Xtensa [Tensilica]. Ο μεταγλωττιστής των επεξεργαστών Xtensa, xt-xcc, είναι μια ενισχυμένη έκδοση της gcc από μια σειρά επεκτάσεων. Όλες οι εφαρμογές-πυρήνες καθώς και 7 από τις 10 εφαρμογές MiBench μεταγλωττίστηκαν επιτυχώς με (-O3) και χωρίς (-O3 -mno-zero-cost-loop) ZOL υποστήριξη και επίσης ενεργοποιώντας (-xpres) και απενεργοποιώντας τη αυτόματη γένεση ειδικών εντολών με την τεχνολογία XPRES.

Τα αποτελέσματα για τις επιδόσεις του LX 1.0 κάτω από τις διαφορετικές αυτές ρυθμίσεις δίνονται στο Σχ. 3.10 σε αντιπαράθεση με τις μετρήσεις για τον ενισχυμένο, κατά ZOLC, XiRisc. Από τις μετρήσεις προκύπτει ότι η επιτάχυνση της εφαρμογής λόγω του ZOL στον Xtensa περιορίζεται σε σχετικά χαμηλά επίπεδα (3-7.5%). Η κύρια αιτία για την οποία συμβαίνει αυτό είναι η ανεπάρκεια του xt-xcc στην ανεύρεση ευκαιριών για εκμετάλλευση του

ZOL όταν οι υποψήφιος, για αυτό, περιοχές κώδικα ενσωματώνουν ροή ελέγχου υπό συνθήκη. Σε αντίθεση με αυτό, στην περίπτωση του ZOLC, ροή ελέγχου υπό συνθήκη υποστηρίζεται άμεσα σε bwd διεργασίες. Σε ορισμένες περιπτώσεις (π.χ. sha), η τεχνική ZOL για τον Xtensa εμφανίζεται να αποδίδει καλύτερα από το ZOLC, αλλά αυτό συμβαίνει λόγω των πιο επιθετικών βελτιστοποιήσεων που διαθέτει ο xt-xcc που έχουν ως αποτέλεσμα την μείωση των κύκλων μηχανής για τα επίμαχα DPT.

Επίσης αναλύθηκαν ορισμένες μικροεφαρμογές δοκιμής για την μελέτη της συμπεριφοράς του ZOLC όσον αφορά εφαρμογές με μη-μειώσιμες περιοχές ροής ελέγχου:

- Μια υλοποίηση του επινοήματος του Duff (duff) παρμένη από την σουίτα WCETbench [WCETbench]
- Τρεις παραλλαγές του βασικού μη-μειώσιμου γράφου ροής ελέγχου που αποτελεί αντικείμενο συζήτησης στο [Jan97] για διαφορετικά μεγέθη των DPT, υλοποιημένα σε γλώσσα συμβολομεταφραστή του MIPS-I.

Καθώς οι υπάρχοντες μεταγλωττιστές που βασίζονται στον gcc δεν βελτιστοποιούν τέτοιες περιοχές, οι στοχευόμενοι επεξεργαστές δεν μπορούν να εκμεταλλευτούν βελτιστοποιήσεις που αφορούν την εκτέλεση εντός τέτοιων βρόχων. Για μικρά μεγέθη των DPT, τα κέρδη στις επιδόσεις από τη χρήση της εκδοχής ZOLCfull είναι ιδιαίτερα σημαντικά, αν δε το μέγεθος του DPT αυξηθεί σε 10 εντολές, ο βαθμός μείωσης των κύκλων ελαττώνεται, παραμένει όμως υπολογίσιμος όπως και προκύπτει μελετώντας τον Πίνακα 3.5.

Πίνακας 3.5: Αποτελέσματα επιδόσεων για περιπτώσεις μη-μειώσιμης ροής ελέγχου.

Benchmark	Description	Cyc. with ZOL (ZOLC if MIPS-I)	Cyc. without ZOL (ZOLC if MIPS-I)	%diff
<i>duff</i>	Duff's device on MIPS-I	1241	1750	29.1
<i>duff</i>	Duff's device on Xtensa LX	1300	1300	0.0
<i>bfg1</i>	Irreducible CFG with DPT size = 1	12	64	81.25
<i>bfg5</i>	Irreducible CFG with DPT size = 5	60	119	49.6
<i>bfg10</i>	Irreducible CFG with DPT size = 10	120	175	31.4

3.5.3. Πειράματα για τον hDSP ASIP

Οι εύκολα επαναστοχευόμενες προδιαγραφές του ZOLC της ενότητας 3.3.3 επαναχρησιμοποιήθηκαν για την σύνταξη ενός προσομοιωτή ακρίβειας κύκλου σε ArchC για τον επεξεργαστή hDSP. Ο hDSP διαθέτει εντολές επέκτασης όπως 'absolute value difference and accumulation' (dabsa) για την επιτάχυνση της εκτέλεσης κρίσιμων, για τις συνολικές επιδόσεις, πυρήνων, όπως είναι για παράδειγμα το κριτήριο SAD στους αλγορίθμους εκτίμησης κίνησης. Η μέση έκταση βασικού μπλοκ και DPT μειώνεται σημαντικά εξαιτίας της χρήσης των εντολών επέκτασης.

Για τον hDSP εξετάστηκε ένα μικρό σύνολο εφαρμογών δοκιμής αποτελούμενο από 2 συνθετικά και 3 πραγματικά εφαρμογίδια, με τα αντίστοιχα αποτελέσματα να απεικονίζονται στον Πίνακα 3.6. Στον Πίνακα 3.6 δίνεται και το μέσο μέγεθος διεργασίας και ο αριθμός κύκλων αρχικοποίησης του ZOLC. Βρέθηκε έτσι, ότι η αύξηση των επιδόσεων κυμαίνεται κατά μέσο όρο στο 44.15% για τα πραγματικά εφαρμογίδια, ενώ επιταχύνσεις του 75% επιτυγχάνονται για τις ακραίες περιπτώσεις (loop4, loop8) της μιας χρήσιμης υπολογιστικά εντολής ανά DPT για τον θεωρούμενο ASIP (εννοείται ότι η ακολουθία εντολών επιβάρυνσης βρόχου χρειάζεται 4 κύκλους μηχανής).

Πίνακας 3.6: Αποτελέσματα επιδόσεων για τα εξετασθέντα εφαρμογίδια στον hDSP.



Benchmark	Avg. task size	Init. cycles	Cyc. with ZOLC	Cyc. without ZOLC	%diff
<i>loop4</i>	1	48	12,786	52,092	75.45
<i>loop8</i>	1	92	1,368,237	5,658,686	75.82
<i>fsme</i>	2.3	58	51,791,075	76,144,025	31.98
<i>matmult</i>	1.6	47	5,184,473	8,992,559	42.35
<i>fsme_dr</i>	2.23	164	20,296,016	48,470,213	58.13

4. ΤΟ ΓΕΝΙΚΟ ΑΡΧΙΤΕΚΤΟΝΙΚΟ ΠΕΡΙΓΡΑΜΜΑ ΤΩΝ ΕΕΣ

Στην ενότητα αυτή, συνοψίζεται το γενικό αρχιτεκτονικό περίγραμμα, και το οποίο παρουσιάζεται στο Σχήμα 4.1. Η ανάλυση των επιμέρους χαρακτηριστικών του γενικού αρχιτεκτονικού περιγράμματος αποτελεί αντικείμενο των κεφαλαίων 2 και 3:

- η αρχιτεκτονική διαύλου για τη διασύνδεση σε επίπεδο συστήματος (ενότητα 2.3)
- η οργάνωση μνήμης και καταχωρητών για τον ΕΕΣ (2.4)
- τα χαρακτηριστικά και η διασύνδεση (με τον ΕΕΣ) των ειδικών λειτουργικών μονάδων (2.5)
- ενίσχυση της λογικής ελέγχου με την αρχιτεκτονική ZOLC (ενότητα 3)

Ειδικότερα στο Σχήμα 4.1.α δείχνεται ένα πρότυπο SoC με αρχιτεκτονική διαύλου AMBA 2.0 AHB. Το SoC διαθέτει έναν ΕΕΣ (που στοχεύει εφαρμογές πολυμέσων, όπως αυτές της σουίτας εφαρμογίδων ZOLC-bench), και ο οποίος επικοινωνεί με αρχιτεκτονικές επεκτάσεις υλικού (ASHEs) σε τρία διαφορετικά επίπεδα (ενότητα 2.5.3):

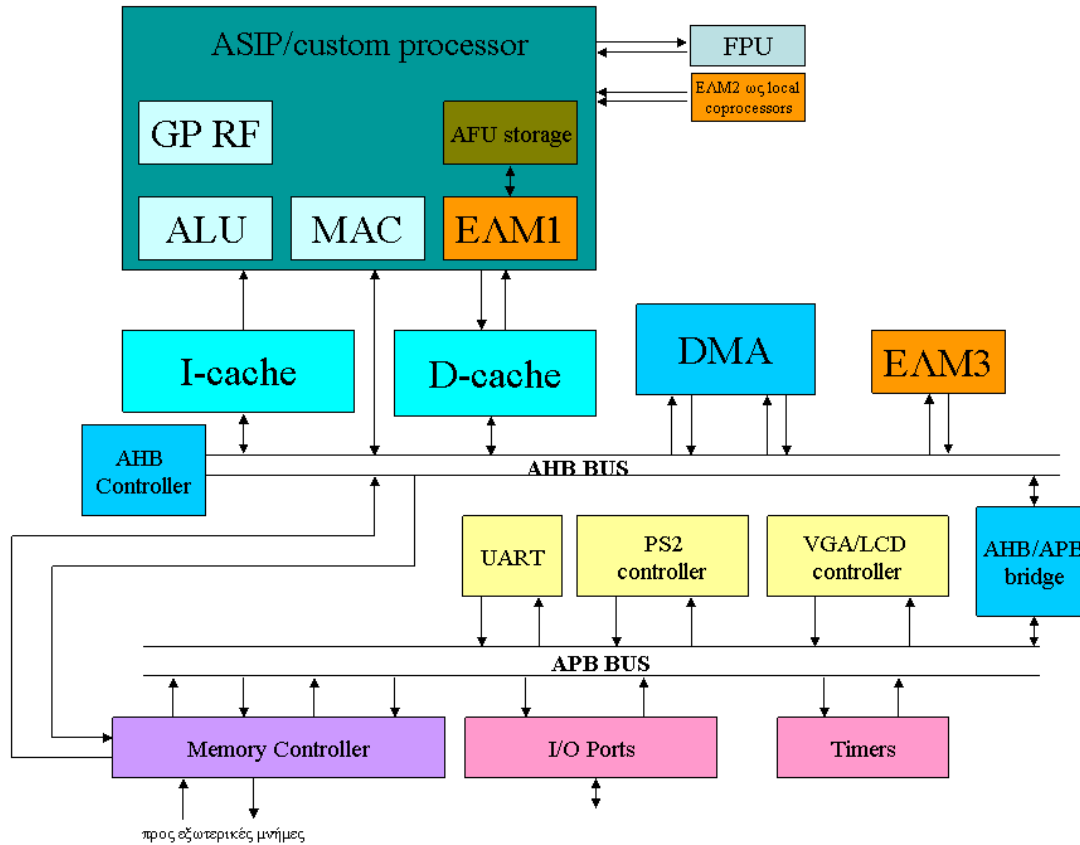
- *σε στενή σύζευξη*, κατά την οποία οι ΕΛΜ ενσωματώνονται στον ΕΕΣ στα στάδια διοχέτευσης του. Οι ΕΛΜ του τύπου αυτού μπορούν να διαθέτουν τοπικούς καταχωρητές, να επικοινωνούν με τη μνήμη δεδομένων με λειτουργίες φόρτωσης/αποθήκευσης ενταγμένες σε ειδικές εντολές. Η αποκωδικοποίηση των ειδικών εντολών για τους ΕΛΜ στενής σύζευξης γίνεται από τον ΕΕΣ. Στο παράδειγμα του Σχήματος 4.1.α ο ΕΛΜ υλοποιεί ένα πολλαπλασιασμό με σταθερά (352x).
- *σε σύζευξη τοπικού συνεπεξεργαστή*, κατά την οποία οι ΕΛΜ επικοινωνούν με τον ΕΕΣ μέσω τοπικής διασύνδεσης (πρωτόκολλο σημείο-σε-σημείο) με ειδικές θύρες εισόδου/εξόδου του ΕΕΣ. Μεταβιβάζονται σήματα ελέγχου από τον ΕΕΣ στον/στους ΕΛΜ ενώ οι θύρες αυτές χρησιμοποιούνται για τη μεταβίβαση ορισμάτων από και προς τους ΕΛΜ. Στο παράδειγμα του Σχ. 4.1.α ως ΕΛΜ θεωρείται μια μονάδα αριθμητικής κινητής υποδιαστολής (FPU).
- *σε χαλαρή σύζευξη μέσω διαύλου*, κατά την οποία οι ΕΛΜ επικοινωνούν μέσω διαύλου. Η περίπτωση αυτή αφορά κυρίως επιταχυντές εφαρμογών τα χαρακτηριστικά των οποίων δεν μεταβάλλονται από πρότυπο σε πρότυπο. Στο αντίστοιχο παράδειγμα του Σχ. 4.1.α δείχνεται μια μονάδα CRC.

Οι ΕΛΜ όλων των κατηγοριών μπορούν να διαθέτουν εσωτερική μικροαρχιτεκτονική με στάδια διοχέτευσης και στοιχεία τοπικής αποθήκευσης.

Η αρχιτεκτονική μνήμης για τον ΕΕΣ τυπικά περιλαμβάνει διαχωρισμένες κρυφές μνήμες για εντολές και δεδομένα, που επικοινωνούν με μια μεγάλη, ενιαία, on-chip μνήμη. Οι κρυφές μνήμες μπορούν να έχουν σχετικά μικρό μέγεθος (από 128 ως 4K bytes), ενώ στην περίπτωση υλοποίησης σε ASIC, η ενιαία μνήμη (με πρόσβαση απλού κύκλου) είναι τυπικά 256Kbytes, ενώ για διατάξεις FPGA το μέγεθος της μνήμης περιορίζεται από τον αριθμό των διαθέσιμων block RAM (κατά την ορολογία των Xilinx FPGA).

Για τον ΕΕΣ επίσης θεωρούμε ότι διαθέτει ένα (αρκετά περιορισμένο μεν) βασικό σύνολο εντολών το οποίο του επιτρέπει να υποστηρίζει όλες τις ακέραιες πράξεις σε γνωστές γλώσσες προγραμματισμού (ANSI C) στο πνεύμα των DLX και SABRE [SABRE]. Ο ΕΕΣ είναι ένας RISC με τον κατάλληλο αριθμό στοιχείων διοχέτευσης. Η οργάνωση των σταδίων διοχέτευσης του μπορεί να είναι ετερογενής με τυπικό μέγιστο αριθμό σταδίων διοχέτευσης από 5 ως 9. Η λογική ελέγχου του ΕΕΣ μπορεί να διαθέτει επιμέρους λογική ZOLC (για την εντατική επεξεργασία δεδομένων σε βρόχους).

Ουσιαστικός στόχος της μεθοδολογίας είναι η επιτάχυνση των στοχευόμενων εφαρμογών με χρήση του ΕΕΣ κατά βαθμό 2x-20x. Για την επίτευξη των μεγαλύτερων δυνατών επιταχύνσεων εφαρμογών θα πρέπει να αξιοποιηθούν ΕΛΜ και στα τρία επίπεδα (Φάσεις 4-6).



Σχήμα 4.1: Γενικό αρχιτεκτονικό περίγραμμα πρότυπου SoC με ΕΕΣ (επεξεργαστή ειδικού σκοπού) της μεθοδολογίας. Ο χρησιμοποιούμενος ΕΕΣ στοχεύει εφαρμογίδα πολυμέσων (σουίτα ZOLCbench).

5. ΓΛΩΣΣΑΡΙ ΟΡΩΝ

Addressing mode	τρόπος διευθυνσιοδότησης
Application analysis	ανάλυση εφαρμογών
Arbitration	διαιτησία
Argument	όρισμα (κατηγορημα) συνάρτησης
Argument stack	στοίβα ορισμάτων
Assembler	συμβολομεταφραστής
Assembly	γλώσσα συμβολομεταφραστή
Backward compatibility	προς-τα-πίσω (αντίρροπη) συμβατότητα
Benchmarking	δοκιμασία επιδόσεων
Bitwidth analysis	ανάλυση εύρους ψηφίου
Burst transfer	μεταβίβαση ριπής
Bus architecture	αρχιτεκτονική διαύλου
Cache analysis	ανάλυση κρυφής μνήμης
Code selection	επιλογή κώδικα
Compiler	μεταγλωττιστής
Compiler infrastructure	υποδομή μεταγλωττιστή
Concatenation	συνένωση
Configuration	ρύθμιση, προσδιορισμός
Consistent data memory	συναφής μνήμη δεδομένων
Control step	βήμα (λογικής) ελέγχου
Critical path	κρίσιμο μονοπάτι
Customizable processor	προσαρμοζόμενος επεξεργαστής
Cycle accurate	ακρίβειας κύκλου
Cycle estimation	εκτίμηση κύκλου
Designed from scratch	σχεδιασμένος εκ του μηδενός
Debugging	εκσφαλμάτωση
Embedded software	ενσωματωμένο λογισμικό
Emulator	εξομοιωτής
Exception	εξάιρεση
File system	σύστημα αρχείων
Graph	γράφος
Graph matching	ταίριασμα γράφων
Hardware	υλικό
Ideal consistent AFU memory	ιδανική συναφής μνήμη ΕΑΜ
Immediate	άμεσο όρισμα
Instruction accurate	ακρίβειας εντολής
Instruction fetch	ανάκληση εντολής
Instruction scheduler	χρονοδρομολογητής εντολών
Interrupt	διακοπή
Intrinsic ILP	εγγενές ILP
Iterated register coalescing	επαναληπτική συνάσπιση καταχωρητών
Linker	συνδέτης
Loop distribution	κατανομή βρόχων
Loop unrolling	ξετύλιγμα βρόχων
Memory consistency	συνάφεια μνήμης
Microkernel	μικροπυρήνας (διαχείρισης λειτουργιών συστήματος)
Module	άρθρωμα
Motion compensation	αντιστάθμιση κίνησης
Motion estimation	εκτίμηση κίνησης
Multimedia	πολυμέσα
Multiplexer	πολυπλέκτης
Node splitting	διαχωρισμός βρόχων
Operand	έντελο (λειτουργίας)
Paging	σελιδοποίηση
Pass	εδώ: πέρασμα μεταγλωττιστή
Pipeline stage	βαθμίδα διοχέτευσης



Place-and-route		τοποθέτηση-και-διασύνδεση
Point-to-point		σημείο-σε-σημείο
Portable		μεταφερτός
Public domain		δημόσιο πεδίο
Retargeting		επαναστόχευση
Register allocator		κατανομητής καταχωρητών
Resource sharing		διαμοιρασμός πόρων
Retargetable compiler		επαναστοχευόμενος μεταγλωττιστής
Scheduling		(χρονο-) δρομολόγηση
Simulator		προσομοιωτής
Software		λογισμικό
Software development toolchain		αλυσίδα εργαλείων ανάπτυξης λογισμικού
Source code		πηγαίος κώδικας
State register		καταχωρητής κατάστασης
Tag		πινακίδα
Task switching		μεταγωγή διεργασιών
Terminal		τερματικό
Virtual memory		εικονική μνήμη
Wrapper		λογική περιτυλίγματος
ADL	Architecture Description Language	Γλώσσα Περιγραφής Αρχιτεκτονικής
ASAP	As Soon As Possible	
ASHE	Application-Specific Hardware Extension	
ASIP	Application-Specific Instruction-set Processor	Επεξεργαστής Ειδικού Σκοπού Συνόλου Εντολών
AFU	Application-specific Functional Unit	Ειδική Λειτουργική Μονάδα
BB	Basic block	Βασικό μπλοκ
CCDG	Composed Control Dependence Graph	Συντεθειμένος Γράφος Εξάρτησης Ελέγχου
CDFG	Control-Data Flow Graph	Γράφος Ροής Ελέγχου-Δεδομένων
CFG	Control Flow Graph	Γράφος Ροής Ελέγχου
CISC	Complex Instruction Set Computer4	Υπολογιστής Σύνθετου Συνόλου Εντολών
CTE	Control Transfer Expression	Έκφραση Μεταφοράς Ελέγχου
CTI	Control Transfer Instruction	Εντολή Μεταφοράς Ελέγχου
DDG	Data Dependence Graph	Γράφος Εξάρτησης Δεδομένων
DMA	Direct Memory Access	Άμεση Προσπέλαση Μνήμης
DPT	Data Processing Task	Διεργασία Επεξεργασίας Δεδομένων
DSE	Design Space Exploration	Διερεύνηση Πεδίου Λύσεων
DSP	Digital Signal Processor/Processing	Επεξεργαστής/Επεξεργασία Ψηφιακού Σήματος
GCC	GNU Compiler-Compiler	
HDL	Hardware Description Language	Γλώσσα Περιγραφής Υλικού
HLL	High Level Language	Γλώσσα Υψηλού Επιπέδου
HLS	High-Level Synthesis	Σύνθεση Υψηλού Επιπέδου
ILP	Instruction-Level Parallelism	Παραλληλία Επιπέδου Εντολής
IR	Intermediate Representation	Ενδιάμεση Αναπαράσταση (μεταγλωττιστή)
LUT	Look-Up Table	Πίνακας Αναζήτησης
MISO	Multiple-Input/Single-Output	Πολλαπλών Εισόδων/Μονής Εξόδου
MIMO	Multiple-Input/Multiple-Output	Πολλαπλών Εισόδων/Πολλαπλών Εξόδων
MMU	Memory Management Unit	Μονάδα Διαχείρισης Μνήμης
NoC	Network-on-chip	Δίκτυο-σε-ολοκληρωμένο
PC	Program Counter	Απαριθμητής Προγράμματος
PDG	Program Dependence Graph	Γράφος Εξάρτησης Προγράμματος
RISC	Reduced Instruction Set Computer	Υπολογιστής Μειωμένου Συνόλου Εντολών
RISP	Reconfigurable Instruction Set Processor	Επαναπροσδιορίσιμος Επεξεργαστής Συνόλου Εντολών
SoC	System-on-Chip	Σύστημα-σε-ολοκληρωμένο
SSA	Static Single Assignment	Στατική Απλή Ανάθεση
SUIFrm	SUIF real machine	Πραγματική Μηχανή SUIF
SUIFvm	SUIF virtual machine	Εικονική Μηχανή SUIF



TCFG	Task-Control Flow Graph	Γράφος Ροής Ελέγχου Διεργασιών
TLM	Transaction-Level Model	Μοντέλο Επιπέδου Συναλλαγής
VLIW	Very Long Instruction Word	Πολύ Μεγάλο Μήκος Εντολής
ZOLC	Zero-Overhead Loop Controller	Ελεγκτής Βρόχων Μηδενικής Επιβάρυνσης
BAM	Basic functional unit	Βασική Λειτουργική Μονάδα
ΕΕΣ	Application-specific Processor	Επεξεργαστής Ειδικού Σκοπού
ΕΑΜ	Application-specific functional unit	Ειδική Λειτουργική Μονάδα

6. ΠΑΡΑΡΤΗΜΑ

A. Περιορισμός κόμβου τύπου-A

Ο περιορισμός κόμβου τύπου-A ή συνοριακού κόμβου [Kan05b] απαγορεύει την ανάπτυξη μιας κατάτμησης γράφου που απαρτίζεται από στοιχειώδεις εντολές (π.χ. SUIFvm) πέραν της σημειωθείσας εντολής. Έχει παρατηρηθεί ότι η εφαρμογή του σε εντολές μεταφοράς δεδομένων (φόρτωση, αποθήκευση, αντιγραφή από μνήμη σε μνήμη) ευνοεί την γένεση ειδικών εντολών που υλοποιούν σύνθετους τρόπους διευθυνσιοδότησης.

B. Περιορισμός κόμβου τύπου-B

Ο περιορισμός κόμβου τύπου-B ή συμπερίληψης κόμβου [Kan05b] δεν επιτρέπει την συμπερίληψη της σημειωθείσας εντολής στην υποψήφια ειδική εντολή που βρίσκεται υπό αναγνώριση. Έχει εφαρμοστεί από τον ερευνητή Α.Π.Θ. σε αλγόριθμο γένεσης εντολών μονής εξόδου (MISO). Συνήθως εφαρμόζεται σε εντολές αλλαγής ελέγχου (control-transfer instructions – cti) όπως είναι οι διακλαδώσεις υπό και χωρίς συνθήκη, και οι λειτουργίες κλήσης και επιστροφής από υποσυνάρτηση.

7. ΑΝΑΦΟΡΕΣ

- [Agg03] A. Aggarwal and M. Franklin, "Energy efficient asymmetrically ported register files," in Proceedings of the 21st International Conference on Computer Design, San Jose, CA, USA, October 2003, pp. 2-7.
- [AHB-Lite] AHB-Lite overview. ARM AMBA Documentation. 2001.
- [Aho86] A. V. Aho, R. Sethi and J. D. Ullman, Compilers: Principles, Techniques, and Tools, Addison-Wesley, Reading, MA, USA, 1986.
- [AMBA] AMBA overview. <http://www.arm.com/products/solutions/AMBAHomePage.html>
- [ARC] ARC homepage. <http://www.arc.com>
- [ArchC] The ArchC resource center. <http://www.archc.org>
- [ARM] ARM homepage. <http://www.arm.com>
- [ASG] AHB System Generator. http://www.opencores.org/projects.cgi/web/ahb_system_generator/overview
- [Aus02] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An infrastructure for computer system modeling," IEEE Computer, Vol. 35, No. 2, pp. 59-67, Feb. 2002.
- [Aze05] R. Azevedo, S. Rigo, M. Bartholomeu, G. Araujo, C. Araujo, and E. Barros, "The ArchC architecture description language and tools," International Journal of Parallel Programming, Vol. 33, No. 5, pp. 453-484, Oct. 2005.
- [Ben88] M. E. Benitez and J. W. Davidson, "A portable global optimizer and linker," in Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation, Atlanta, USA, 1988, pp. 329-338.
- [Beu04] J.-L. Beuchat. A VHDL Library for Integer and Modular Arithmetic - User Manual, 0.1 edition, Sept. 2004. <http://perso.ens-lyon.fr/jean-luc.beuchat/ArithLib>
- [Binutils] GNU Binutils. <http://sources.redhat.com/binutils/>
- [Bis07] P. Biswas, N. L. Dutt, L. Pozzi, and P. lenne, "Introduction of architecturally visible storage in instruction set extensions," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 26, No. 3, pp. 435-446, March 2007.
- [Bro04] A. Brown, S. Z. Guyer, and D. A. Jimenez, "The C-Breeze compiler infrastructure," Department of Computer Sciences, The University of Texas, Austin, TX, Technical report, July 2004.
- [Bur97] D.C. Burger and T.M. Austin, "The SimpleScalar tool set, version 2.0", Technical report CS-TR-1997-1342, Computer Sciences Department, University of Wisconsin-Madison, Jun. 1997.
- [Cam01] F. Campi, R. Canegallo, and R. Guerrieri, "IP-reusable 32-bit VLIW RISC core," in Proceedings of the 27th European Solid-State Circuits Conference, Villach, Austria, September 2001, pp. 456-459.
- [Cat98] F. Catthoor, S. Wuytack, E. De Greef, F. Balasa, L. Nachtergaele, A. Vandecapelle, *Custom Memory Management Methodology: Exploration of Memory Organization for Embedded Multimedia System Design*, Kluwer Academic Publishers, Boston, 1998.
- [Cla03] N. Clark, H. Zhong, W. Tang, and S. Mahlke, "Automatic design of application specific instruction set extensions through dataflow graph exploration," International Journal of Parallel Programming, Vol. 31, No. 6, pp. 429-449, Dec. 2003.
- [Cla05] N. Clark, J. A. Blome, M. L. Chu, S. A. Mahlke, S. Biles, and K. Flautner, "An architecture framework for transparent instruction set customization in embedded processors," in *Proc. 32nd Int. Symp. on Computer Architecture*, Madison, Wisconsin, USA, June 2005, pp. 272-283.
- [CoreConnect] The Official IBM CoreConnect Website. <http://www-306.ibm.com/chips/products/coreconnect/>
- [Ertl99] M. A. Ertl, "Optimal code selection in DAGs," ACM Symposium on Principles of Programming Languages (POPL'99).
- [Eyr00] J. Eyre and J. Bier, "Evolution of DSP processors," *IEEE Signal Processing Magazine*, vol. 17, no. 2, pp. 43-51, March 2000.
- [Fer87] J. Ferrante, K. J. Ottenstein and J. D. Warren, "The program dependence graph and its use in optimization," ACM Transactions on Programming Languages and Systems, Vol. 9, No. 3, pp. 319-349, July 1987.
- [Fly97] David Flynn, "AMBA: Enabling Reusable On-chip Designs", IEEE Micro, Vol 17, No. 4, July-August 1997, pp. 20-27.



- [Fri04] J. Fritts. MediaBench II. In 2004 Workshop on Media and Signal Processors for Embedded Systems and SoCs, pp. 723–728, Sept. 2004.
- [FSL] Xilinx, Fast Simplex Link (FSL) Bus (v2.00a), Product specification DS 449, December 1, 2005.
- [Gaisler] Gaisler research. <http://www.gaisler.com>
- [GCC] The GNU Compiler Collection homepage. <http://gcc.gnu.org>
- [Geo96] L. George and A. W. Appel, “Iterated register coalescing,” ACM Transactions on Programming Languages and Systems, vol. 18, no. 3, pp. 300.324, May 1996.
- [Geu05] W. Geurts, G. Goossens, D. Lanneer, J. Van Praet, “Design of Application-Specific Instruction-Set Processors for Multi-Media, using a Retargetable Compilation Flow,” Proceedings of GSPx 2005.
- [Gon00] R. Gonzalez, “Xtensa: A configurable and extensible processor,” *IEEE Micro*, vol. 20, no. 2, pp. 60-70, March-April 2000.
- [Greenbus] GreenBus homepage. <http://www.greensocs.com/GreenBus/>
- [GRFPU] GRFPU High-Performance Floating-Point Unit. <http://www.gaisler.com>
- [Gri03] M. Gries, “Methods for Evaluating and Covering the Design Space during Early Design Development,” Technical Memorandum, UCB/ERL M03/32, CAD-Group, Electronics Research Laboratory, University of California at Berkeley, CA, USA, Aug. 2003.
- [GXemul] Anders Gavare. GXemul homepage. <http://gavare.se/gxemul/>
- [Hen96] J.L. Hennessy, and D.A. Patterson, Computer Architecture: a Quantitative Approach, 2nd edition, Morgan Kaufmann Publishers, San Fransisco, CA, 1996.
- [Hof02] A. Hoffmann, H. Meyr, and R. Leupers, Architecture exploration for embedded processors with LISA, Kluwer Academic Publishers, Dec. 2002.
- [HWLU] N. Kavvadias. Hardware looping unit. <http://www.opencores.org/projects.cgi/web/hwlu/overview/>
- [IBURG] IBURG, a tree parser generator. <http://www.cs.princeton.edu/software/iburg/>
- [Jan97] J. Janssen and H. Corporaal, “Making graphs reducible with controlled node splitting,” ACM Transactions on Programming Languages and Systems, vol. 19, no. 6, pp. 1031.1052, November 1997.
- [JOP] Martin Schoeberl, “JOP: A Java-Optimized Processor”. <http://www.jopdesign.com>
- [Kav05a] N. Kavvadias and S. Nikolaidis, “Zero-overhead loop controller that implements multimedia algorithms,” IEE Proceedings - Computers and Digital Techniques, vol. 152, no. 4, pp. 517.526, July 2005.
- [Kav05b] N. Kavvadias and S. Nikolaidis, “Automated Instruction-Set Extension of Embedded Processors with Application to MPEG-4 Video Encoding,” in Proceedings of the IEEE 16th International Conference on Application-specific Systems, Architectures and Processors, pp. 140-145, July 2005, Samos, Greece.
- [Kav06] N. Kavvadias and S. Nikolaidis, “A portable specification of zero-overhead looping control hardware applied to embedded processors,” in Proceedings of the 2006 IEEE International Symposium on Circuits and Systems, pp. 1599-1602, May 21-24, 2006, Kos, Greece.
- [Kod03] Varun Kodthivada (1999216), Prashant Aggarwal (1999201), “Automated Design of a Functionally Pipelined Co-Processor,” B.Sc. thesis, Department of Electrical Engineering, Indian Institute of Technology at Delhi, Delhi, India, May 2003.
- [Kum00] S. Kumar, L. Pires, S. Ponnuswamy, C. Nanavati, J. Golusky, M. Vojta, S. Wadi, D. Pandalai, and H. Spaanenbunrg, “A benchmark suite for evaluating configurable computing systems - status, re_ections, and future directions,” in Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, CA, USA, February 2000.
- [LEON2-CIS] LEON2-CIS project homepage. http://www.iaik.tugraz.at/research/vlsi/01_projects/01_isec/05_Leon2-CIS/index.php
- [Leu03] R. Leupers, O. Whalen, M. Hohenauer, T. Kogel, and P. Marwedel, “An executable intermediate representation for retargetable compilation and high-level code optimization,” Proceedings of the 1st International Workshop on Systems, Architectures, Modeling, and Simulation, pp. 120-125, July 2003, Samos, Greece.
- [Leu06] R. Leupers, K. Karuri, S. Kraemer, and M. Pandey, “A design flow for configurable embedded processors based on optimized instruction set extension synthesis,” in *Proc. of the Design, Automation and Test in Europe Conf.*, March 6-10 2006.
- [Lui06] Ilie I. Luican, Hongwei Zhu, and Florin Balasa, “Formal Model of Data Reuse Analysis for Hierarchical Memory Organizations,” in Proc. of International Conference on Computer-Aided Design (ICCAD’06), Nov. 5-9, 2006, San Jose, CA, USA.
- [MachSUIF] Machine-SUIF research compiler. <http://www.eecs.harvard.edu/hube/research/machsuiif.html>



- [MiBench] MiBench benchmark suite. <http://www.eecs.umich.edu/mibench/>
- [MicroBlaze] MicroBlaze Soft Processor Core. <http://www.xilinx.com>
- [MIPS] MIPS Technologies Inc.. <http://www.mips.com>
- [Mlite] Plasma – most MIPS-I™ opcodes overview. <http://www.opencores.org/projects.cgi/web/mips/overview>
- [Mom06] S. Momcilovic, T. Dias, N. Roma, L. Sousa, “Application Specific Instruction Set Processor for Adaptive Video Motion Estimation,” Proceedings of the 9th EUROMICRO Conference on Digital System Design (DSD’06).
- [MOVE] ARM Ltd., MOVE Coprocessor Technical Reference Manual, April 2002.
- [MPEG4-VVM18] MPEG-4 video verification model version 18.0. MPEG-4 draft standard, International Organization of Standardization, Working Group on Coding of Moving Pictures and Audio, Pisa, Italy, Jan. 2001.
- [MPEG4senc] MPEG-4 shape encoder project. <http://www.ece.cmu.edu/~ece796/project99/12/final/>
- [Nios-II] Altera Nios-II home page. <http://www.altera.com/products/ip/processors/nios2/>
- [Nios-II-CI] Altera Corporation, “Nios II Custom Instruction: User Guide,” December 2004.
- [Nios-II-DCB] Nios II Embedded Processor Design Contest book, Outstanding Designs 2005.
- [NoCem] Network-on-Chip emulator. <http://www.opencores.org/projects.cgi/web/nocem/>
- [Noxim] Network-on-Chip simulator. <http://sourceforge.net/projects/noxim/>
- [OC-CORDIC] CORDIC core. <http://www.opencores.org/projects.cgi/web/cordic/overview>
- [OC-DES] DES/Triple DES IP Cores. <http://www.opencores.org/projects.cgi/web/des/overview>
- [Opencores] Opencores website. <http://www.opencores.org>
- [OpenRISC] OpenRISC processor family homepage. <http://www.opencores.org/projects.cgi/web/or1k/overview>
- [Pan03] C. Pan, N. Bagherzadeh, A. H. Kamalizad, and A. Koohi, “Design and analysis of a programmable single-chip architecture for DVB-T base-band receiver,” in Proceedings of the Design, Automation and Test in Europe Conference, Munich, Germany, March 2003, pp. 468-473.
- [Pel03] P. Pelgrims, T. Tierens and D. Driessens, “Overview of Embedded Busses,” version 1.1.
- [Qemu] Fabrice Bellard. Qemu homepage. <http://www.qemu.org>
- [Ric03] N. Richardson, L.B. Huang, R. Hossain, J. Lewis, T. Zounes and N. Soni, “The ICORE 520-MHz synthesizable CPU core,” IEEE Micro, Vol. 23, No. 3, pp. 46-57, May-June, 2003.
- [Roh96] E. Rohou, F. Bodin, A. Sez nec, G. L. Fol, F. Charot, and F. Raimbault, “SALTO: System for assembly-language transformation and optimization,” Institut National de Recherche en Informatique et en Automatique, Technical report 2980, September 1996.
- [SABRE] SABRE RISC processor example. <http://www.celoxica.com>
- [Sag07] Mazen A. R. Saghir and Rawan Naous, “A Configurable Multi-Ported Register File Architecture for Soft Processor Cores,” *Proceedings of the 2007 International Workshop on Applied Reconfigurable Computing (ARC 2007)*, pp. 14-25, Springer-Verlag LNCS 4419, March 27-29, 2007.
- [SALTO] SALTO. <http://www.irisa.fr/caps/projects/Salto/>
- [Schaum03] P. Schaumont and I. Verbauwhede, “Domain-Specific Codesign for Embedded Security,” IEEE Computer, April 2003, pp. 68-74.
- [She04] T. P. Shevlin, “Composed Control Dependence Graph Generator,” M.Sc. thesis, Department of Computer Science, Tufts University, Medford, Massachusetts, USA, Sep. 2004.
- [Simplebus] R. Hilderink, M. Janssen, T. Groetker and H. Keding, “The Simplebus Abstract Bus Model”, included in the SystemC-2.0.1 distribution. <http://www.systemc.org>
- [SimpleScalar] SimpleScalar simulation framework homepage. <http://www.simplescalar.com>
- [Sin03] A. Singh, A. Chhabra, A. Gangwar, B.K. Dwivedi, M. Balakrishnan, and A. Kumar, “SoC Synthesis with Automatic Hardware Software Interface Generation,” in Proceedings of the 16th International Conference on VLSI Design (VLSI Design 2003), pp. 585-590, 4-8 January 2003, New Delhi, India.
- [Sir07] S. Sirowy, Y. Wu, S. Lonardi, and F. Vahid, “Two-level microprocessor accelerator partitioning,” in Proc. of the Design, Automation and Test in Europe Conf., Nice, France, April 16-20, 2007.
- [Smi02] M.D. Smith and G. Holloway, “An introduction to Machine SUIF and its portable libraries for analysis and optimization,” Technical report, 2.02.07.15 edition, Division of Engineering and Applied Sciences, Harvard University, USA, 2002.



- [SoftFloat] SoftFloat. <http://www.jhauser.us/arithmetic/SoftFloat.html>
- [Sta02] R. M. Stallman, GNU Compiler Collection Internals, for gcc 3.3.2 ed., Free Software Foundation, Inc., Cambridge, Massachusetts, December 2002. <http://gcc.gnu.org/onlinedocs/gccint/>
- [SUIF] SUIF compiler infrastructure. <http://suif.stanford.edu/suif/suif2/>
- [SystemC] SystemC community homepage. <http://www.systemc.org>
- [Tal03] D. Talla, L. K. John, and D. Burger, "Bottlenecks in multimedia processing with SIMD style extensions and architectural enhancements," IEEE Transactions on Computers, vol. 52, no. 8, pp. 1015-1031, August 2003.
- [Tensilica] Xtensa configurable processors. <http://www.tensilica.com>
- [Tou05] Sid-Ahmed-Ali Touati, "Register Saturation in Instruction Level Parallelism," International Journal of Parallel Programming, pp. 393-449, Vol. 33, No. 4, August 2005.
- [Trimaran] Trimaran Consortium, The Trimaran Compiler Research Infrastructure for Instruction Level Parallelism. <http://www.trimaran.org>
- [uClinux] uClinux: Embedded Linux/Microcontroller Project. <http://www.uclinux.org>
- [UCRC] Ultimate CRC. http://www.opencores.org/projects.cgi/web/ultimate_crc/overview
- [Ung02] S. Unger and F. Mueller, "Handling irreducible loops: Optimized node splitting vs. DJ-graphs," *ACM Transactions on Programming Languages and Systems*, vol. 24, no. 4, pp. 299-333, 2002.
- [Uss01] R. Usselman, OpenCores SoC Bus Review, Rev. 1.0, Jan. 9 2001.
- [VCG] VCG website. <http://rw4.cs.ui-sb.de/~sander/html/gsvcg1.html>
- [Via03] P. Viana, E. Barros, S. Rigo, R. Azevedo, and G. Araujo, "Exploring Memory Hierarchy with ArchC," Proc. of the 15th Symp. on Computer Architecture and High Performance Computing, Sao Paulo, (SBAC-PAD'03), November 2003.
- [VMIPS] Vmips emulator. <http://www.dgate.org/vmips/>
- [WCETbench] WCET project benchmarks. <http://www.mrtc.mdh.se/projects/wcet/benchmarks.html>
- [Wishbone] Wishbone System-on-Chip (SoC) interconnection architecture for portable IP cores, revision: B.3. <http://www.opencores.org/projects.cgi/web/wishbone/wishbone>
- [Xilinx] Xilinx homepage. <http://www.xilinx.com>